ACADEMIC RESEARCH BY IBRAHIM SHNOUDI

Separation at Genus Two for the Type II Codes of Length 24: a Greene-Type Specialisation and the Separation Lattice of Genus-g Tutte Polynomials

Ibrahim Amin Nicola Shnoudi

Independent researcher, Amman, Jordan

Email: Ibrahim.shnoudiest@gmail.com · ORCID: 0009-0002-3706-6321

Abstract

Miezaki, Oura, Sakuma and Shinohara introduced the Tutte polynomial of genus g of a matroid. They observe that no relation is known between the complete weight enumerator of genus g of a code and the Tutte polynomial of genus g of its matroid, and their Remark 5.5 asks whether the genus-g Tutte polynomials of the Type II codes of length 8m are linearly independent. We give a partial answer to the first question and settle the second for length 24.

A Greene-type specialisation. For a self-dual binary [n,k] code, the genus-g Whitney rank generating function of its vector matroid, restricted to the subvariety on which every product u_a v_a equals 2, is an explicit specialisation of the complete weight enumerator of genus g^2. For g = 1 this reduces to Greene's identity in the self-dual binary case; the case g = 1 of the reconstruction below is a known equivalence of Britz [7], and the contribution here is the extension to genus g. Interpolating over the family of such subvarieties recovers the whole genus-g function from complete weight enumerators of genus at most $k \cdot g^2$, and at genus two a change of variables lowering the total degree sharpens this to k. The relation is not a same-genus one, and answers Remark 5.5 only in the form $g \mapsto g^2$.

Length 24. The least separating genus is exactly two. The nine genus-one Tutte polynomials span a space of dimension eight, so their relation space is generated by a primitive integer vector w, which we give in closed form as a divided difference in the number of weight-four codewords; the genus-two relation space is contained in the genus-one one by an elementary specialisation; and an exact finite-field evaluation at explicit small integers shows that w is not a genus-two relation. No randomness enters the argument.

The separation lattice. The local weight in the coordinate-wise factorisation depends on the pattern of a tuple of codewords only through the intersection of the state sets that the pattern activates, so the genus-g polynomial sees a complete weight enumerator only through the counts of coordinates in each member of the resulting intersection lattice $\mathcal{L}_g$. Its size is the sum over r of C(g,r)·2^C(r,2), the number of labelled graphs on at most g

vertices. The passage from the weight enumerator to the genus-g polynomial factors into that count followed by a substitution of local weights; the local weights span a space of dimension exactly 2^g, so the second stage is governed by $|\mathcal{L}_g| - 2^g$ independent linear relations, for which we give an explicit basis indexed by the members with non-empty graph. This reduces the first stage of the separation problem to a finite state space determined by the index structure alone, and yields a necessary condition for separation at a given genus.

Algorithms. The genus-one function is computed in $O(n \cdot 2^n)$ operations by a subset-sum transform; the genus-two function is evaluated exactly at any point of a seven-dimensional subvariety, also in $O(n \cdot 2^n)$; and the unrestricted eight-variable genus-two polynomial is evaluated in $O(3^n)$, the exponent 3 arising because intersection and union are closed in opposite directions in the subset lattice. All arithmetic used in the reported computations is integer, rational or finite-field. Throughout, linear independence and relation spaces are taken over the rationals unless another field is stated.

Keywords: separation lattice; Tutte polynomial of genus g; Type II code; Witt's problem; complete weight enumerator; Greene's theorem; Möbius transform.

MSC 2020: 05B35, 94B05, 11T71.

Running head: Separation at genus two for the Type II codes of length 24.

How to verify the computational claims. The file of Appendix A, run on the nine generator matrices of Appendix B, re-derives and asserts every numerical value stated in Sections 5, 7, 8 and 9. Readers preferring not to execute anything may check the certificate of Theorem 18 by hand: the vector w and the identity $\Theta^{T}w = 0$ against the matrix printed in Appendix C, the 8 × 8 minor of Theorem 6, and the single residue of Lemma 17.

1. Introduction

Let M = (E, $\mathcal{I}$) be a matroid with rank function ρ. Its Tutte polynomial is

T(M; x, y) = Σ over A ⊆ E of (x−1)^(ρ(E)−ρ(A)) · (y−1)^(|A|−ρ(A)).

Miezaki, Oura, Sakuma and Shinohara [1, 2] introduced a family of generalisations indexed by a genus parameter g. Set $\Lambda_1 := \{1, \ldots, g\}$ and let Λ_2 be the set of two-element subsets of Λ_1. For a g-tuple $(A_1, \ldots, A_g)$ of subsets of E, the polynomial records each A_i, each pairwise intersection and each pairwise union. We give the definition in full in Section 2, since every later argument depends on it.

The motivating question is a Tutte analogue of a problem of Witt. A binary code is of Type II if it is self-dual and every codeword has weight divisible by four; such codes exist precisely in lengths divisible by eight. **Witt's problem** asks, for each m, for the least g such that the genus-g weight enumerators of the inequivalent Type II codes of length 8m are linearly independent. Nebe [3] determined the answer for m = 2, 3, 4 to be 3, 6 and 10 respectively. The letter n is reserved for the length of a code throughout the rest of the paper.

For the Tutte analogue only genus one had been examined. Miezaki et al. proved by exhaustive enumeration that the genus-one Tutte polynomials of the Type II codes of length 8m are linearly independent for m = 2 and linearly dependent for m = 3, and their Remark 5.5 poses the general determination as an open problem.

The present paper settles the case m = 3 and supplies an explicit bridge between the two sides of the theory. To the best of our knowledge, this is the first determination of the least genus for the Type II codes of length 24.

m (length 8m)	length	codes	least genus for linear independence of weight enumerators	least genus for linear independence of Tutte polynomials
2	16	2	3	1
3	24	9	6	**2** (Theorem 18, below)
4	32	85	10	open

Architecture of the proof of the second result. Write Rel(F) for the space of linear relations among a family F of polynomials. We prove, in this order,

(i) $\mathrm{Rel}(T^{(2)}) \subseteq \mathrm{Rel}(T^{(1)})$, (ii) $\mathrm{Rel}(T^{(1)}) = \langle w \rangle$, (iii) $w \notin \mathrm{Rel}(T^{(2)})$,

from which $\mathrm{Rel}(T^{(2)}) = 0$ follows at once. Step (i) is an elementary specialisation, step (ii) is an exact rational computation certified by a null vector together with a nonvanishing minor, and step (iii) is a single exact evaluation over a finite field. No step depends on randomness or on floating-point arithmetic.

Plan of the paper. Section 2 fixes the definitions, including the genus-g Tutte polynomial in full, and records two lemmas used repeatedly thereafter. Section 3 gives the genus-one algorithm. Section 4 constructs the nine codes of length 24 and fixes the ordering used in every subsequent table, vector and certificate. Section 5 carries out step (ii). Section 6 proves the Greene-type specialisation; it is independent of the length-24 computation and may be read on its own. Section 7 carries out steps (i) and (iii), proves the main theorem, and closes with an algorithm for the unrestricted eight-variable polynomial. Section 8 introduces the separation lattice and the two-stage factorisation it induces, Section 9 applies these at length 24, and Section 10 places the results relative to the known invariant theory of self-dual codes. Theorem 18 is proved in Section 7 and is unconditional on Sections 8 to 10. Sections 11 and 12 collect the open problems and the computational details, and the two appendices contain the reference implementation and the generator matrices.

One comparison should be stated at the outset. The two Type II codes of length 16 have identical weight enumerators, so genus three is required to separate them, whereas their genus-one Tutte polynomials already differ; on that family the genus-one Tutte polynomial is strictly finer. Linear dependence of a family of polynomials is weaker than failure to separate its members pairwise, and all statements below are about linear independence only.

2. Definitions and preliminaries

This section is self-contained: every object used below is defined here, so that the reader need not consult [1] for either the statements or the proofs.

Definition 1 (Whitney rank generating function of genus g). Let M be a matroid on E with rank function ρ. Introduce variables u_i, v_i for i ∈ Λ_1 and u_{ij}^∩, v_{ij}^∩, u_{ij}^∪, v_{ij}^∪ for {i,j} ∈ Λ_2. Then

$R^{(g)}(M)$ = Σ over $(A_1,\dots,A_g) \in (2^E)^g$ of $\prod_{i \in \Lambda_1} u_i^{\rho(E)-\rho(A_i)} \cdot v_i^{|A_i|-\rho(A_i)} \cdot \prod_{\{i,j\} \in \Lambda_2} u_{ij}^{\cap(\rho(E)-\rho(A_i\cap A_j))} \cdot v_{ij}^{\cap(|A_i\cap A_j|-\rho(A_i\cap A_j))} \cdot u_{ij}^{\cup(\rho(E)-\rho(A_i\cup A_j))} \cdot v_{ij}^{\cup(|A_i\cup A_j|-\rho(A_i\cup A_j))}$.

The Tutte polynomial of genus g is obtained by the substitution u ↦ x−1, v ↦ y−1 in every variable:

$T^{(g)}(M) = R^{(g)}(M)$ evaluated at $u_\bullet = x_\bullet - 1$, $v_\bullet = y_\bullet - 1$.

The number of sets appearing is

$g + 2\cdot C(g,2) = g^2$.

For g = 1 the definition reduces to the classical Whitney rank generating function and Tutte polynomial. For g = 2 we abbreviate the four index slots as a ∈ {1, 2, ∩, ∪} and write (u_a, v_a), $t_a := u_a \cdot v_a$.

Lemma 2. *$T^{(g)}$ and $R^{(g)}$ are related by an invertible affine change of variables. Consequently, for any finite family of matroids, the space of linear relations among their genus-g Tutte polynomials equals the space of linear relations among their genus-g Whitney rank generating functions.*

Proof. The substitution u ↦ x − 1, v ↦ y − 1 is an invertible change of polynomial indeterminates, and it is applied identically to every member of the family; a linear combination vanishes identically before the substitution if and only if it vanishes identically after it. ■

All computations below are performed with $R^{(g)}$, and Lemma 2 transfers every conclusion to $T^{(g)}$ without further comment.

Definition 3 (complete weight enumerator of genus g). For a binary code C of length n,

cwe_$C^{(g)}$(x_a: a ∈ F_2^g) = Σ over (c_1,…,c_g) ∈ C^g of $\prod_{a \in F_2^g}$ $x_a^{n_a(c_1,\dots,c_g)}$,

where n_a counts the coordinates i at which the column vector ((c_1)_i, …, (c_g)_i) equals a.

Notation. For A ⊆ E = {1,…,n} let F_2^A denote the set of vectors of F_2^n whose support is contained in A, and put

$d(A) = \dim(C \cap F_2^A)$, $N(A) = \#\{c \in C : \mathrm{supp}(c) \subseteq A\}$.

Since the codewords supported in A form a subspace, N(A) = 2^(d(A)).

Lemma 4. *If C is self-dual of length n and dimension k = n/2, and M(C) is the vector matroid of the columns of a generator matrix, then ρ(E) = k and*

ρ(A) = |A| − d(A) for every A ⊆ E.

Proof. Let π_A: C → F_2^A be the coordinate projection onto A. A codeword lies in its kernel exactly when its support avoids A, so ker π_A = C ∩ F_2^(E\A) and

ρ(A) = rank(G_A) = dim π_A(C) = k − dim(C ∩ F_2^(E\A)).

For the standard nondegenerate bilinear form on F_2^n one has ($F_2^{A)}$⊥ = F_2^(E\A), hence

(C ∩ $F_2^{A)}$⊥ = C^⊥ + F_2^(E\A) = C + F_2^(E\A),

using C = C^⊥. Taking dimensions, and applying to the right-hand side the dimension formula dim(U + V) = dim U + dim V − dim(U ∩ V),

n − d(A) = k + |E\A| − dim(C ∩ F_2^(E\A)),

so dim(C ∩ F_2^(E\A)) = |E\A| − k + d(A), using n = 2k. Substituting into the first display gives ρ(A) = k − |E\A| + k − d(A) = |A| − d(A). Taking A = E gives ρ(E) = k. ■

For a family f_1, …, f_m of polynomials, Rel(f) denotes the space of vectors c with Σ c_i f_i = 0 identically.

3. An O(n·2^n) algorithm at genus one

Definition 1 is a sum over all 2^n subsets, and a direct implementation computes a matrix rank for each of them. The following proposition removes those rank computations entirely, and is what makes the length-24 computation of Section 5 routine.

Proposition 5. *For a self-dual binary code of length n and dimension k, the function d, hence by Lemma 4 the entire rank function of M(C), is computed for all 2^n subsets simultaneously in n·2^(n−1) integer additions and O(2^n) space, using exact integer arithmetic only.*

Proof. Let f be the function on subsets that assigns to a set the number of codewords whose support equals that set; f is obtained in O(|C|·n) operations. Then N is the subset-sum, or zeta, transform of f over the Boolean lattice,

```
for j = 0 .. n-1:
    for all masks m with bit j set:  N[m] += N[m XOR 2^j]
```

which performs n·2^(n−1) additions. Since N(A) = 2^(d(A)) is a power of two bounded by 2^k, the exponent is recovered by a table lookup indexed by N(A), with no logarithm and no floating-point operation. ■

The coefficient list of $R^{(1)}$ is then obtained by tabulating the pairs (ρ(E) − ρ(A), |A| − ρ(A)) over all A. Both entries lie in {0, …, k}, so the pair is encoded without collision as (ρ(E) − ρ(A))·(k+1) + (|A| − ρ(A)).

Validation. At length 16 direct enumeration is inexpensive and both methods agree. The two Type II codes of that length differ in exactly thirteen coefficients; for instance the coefficient indexed by (0,0) equals 3136 for $e_8 \oplus e_8$ and 3712 for d_{16}^+. This reproduces the case n = 2 of the theorem of Miezaki et al. and serves as a validation of the implementation; it is not used in any proof below.

4. The nine Type II codes of length 24

The algorithm of Section 3 takes generator matrices as input. This section supplies them and fixes the ordering of the nine codes that is used in every coefficient vector, table and certificate below.

Throughout, A^⊕m denotes the direct sum of m copies of A.

Representatives were constructed by the accompanying program rather than imported from a database. For G = [I_{12} | A] in standard form, self-duality is equivalent to $A \cdot A^T = I$ over F_2, and double evenness forces every row of A to have weight 3 modulo 4. A randomised search over such matrices, with two explicit constructions added, produced nine distinct isomorphism types.

On exhaustiveness. The classification is classical and is not claimed here; this section contributes only explicit representatives. The randomised search has no logical role: exhaustiveness comes from Pless and Sloane [5], by which there are exactly nine inequivalent doubly even self-dual codes of length 24. The invariant below is preserved under coordinate permutations, so the nine distinct profiles establish pairwise inequivalence.

On the two explicit constructions. In tens of thousands of trials the randomised search never produced e_8^⊕3 or d_{24}^+. The first is the direct sum of three copies of the extended Hamming code. For the second, partition the twenty-four coordinates into twelve disjoint pairs P_1, …, P_{12}; take the eleven vectors supported on $P_1 \cup P_i$ for i = 2, …, 12, each of weight four, together with the transversal vector of weight twelve containing exactly one coordinate of each pair. Any two of the first eleven meet in the two coordinates of P_1, and the transversal meets each of them in exactly two coordinates, so the twelve vectors are pairwise orthogonal and each has weight divisible by four; they are independent, so they generate a doubly even self-orthogonal code of dimension twelve in F_2^{24}, which is therefore self-dual.

The representatives are separated by the pair consisting of A_4, the number of codewords of weight four, and the multiset of connected components of the graph induced on the coordinates by the supports of the weight-four codewords, each component tagged with the number of weight-four codewords it carries.

Table 1. The ordering used for all coefficient vectors in this paper.

i	A_4	component profile (size, weight-4 words)	code
1	0	twenty-four isolated points	g_{24}
2	6	(4,1)^6	d_4^⊕6

i	A_4	component profile (size, weight-4 words)	code
3	12	(6,3)^4	d_6^⊕4
4	18	(8,6)^3	d_8^⊕3
5	24	$(7,7)^2$, (10,10)	$d_{10} \oplus e_7$^⊕2
6	30	$(12,15)^2$	d_{12}^⊕2
7	42	(8,14)^3	e_8^⊕3
8	42	(8,14), (16,28)	$d_{16} \oplus e_8$
9	66	(24,66)	d_{24}^+

For the d_{2m} components occurring here, a component of size 2m carries C(m,2) codewords of weight four, as the data reproduce for m = 2, 3, 4, 6, 12. The two components of size seven in code 5 are e_7 components; their odd size is what identifies them. The nine representatives have pairwise distinct profiles, so they are pairwise inequivalent; by the classification there are exactly nine classes, so they are a full set of representatives. The component decompositions recorded in the last column are the standard names attached to those decompositions, and are stated for orientation only: no argument in this paper depends on them. We do not claim that the profile is a complete invariant for arbitrary Type II codes.

The nine generator matrices used in every computation of this paper are listed in Appendix B, so that all results can be reproduced without rerunning the search. Since coordinate permutations induce matroid isomorphisms, any equivalent choice of representatives yields the same polynomials.

5. The genus-one polynomials: rank and the unique relation

With Proposition 5 and the matrices of Section 4 in place, we carry out step (ii) of the architecture set out in Section 1.

Let X be the union of the monomial supports of the nine polynomials $R_1^{(1)}$, …, $R_9^{(1)}$, in the ordering of Table 1; then |X| = 76. Order X lexicographically by the exponent pair (ρ(E) − ρ(A), |A| − ρ(A)), and let Θ be the 9 × 76 integer matrix whose i-th row is the coefficient vector of $R_i^{(1)}$ in this ordered basis. No coefficient is omitted.

Theorem 6 (exact computational theorem). *The matrix Θ has rank exactly 8 over ℚ. Hence Rel($R^{(1)}$), and by Lemma 2 also Rel($T^{(1)}$), is one-dimensional, generated by the primitive integer vector*

w = (576, −2772, 4928, −3465, 0, 924, 99, −297, 7).

The rank assertion is not the output of a black-box rank routine: it is certified by the displayed vanishing relation together with the displayed nonzero minor, either of which the reader may check independently.

Proof. The vector w satisfies $\Theta^{T}w = 0$ exactly, verified in all 76 coordinates; this gives rank $\Theta \le 8$. For the reverse inequality we exhibit a nonvanishing 8×8 minor. Take the first eight rows of Table 1 and the eight columns of X with exponent pairs

(0,8), (3,6), (4,6), (5,2), (5,4), (5,5), (8,2), (12,0).

The resulting integer submatrix is

```
10626      0     0      0     0    0    0   1
10620      0     0    240     0    0    0   1
10614      0     0   4128     0    0    4   1
10608     48     0  11328     0    0   12   1
10602    352     1  20384    48    0   24   1
10596    480    12  34320   168    2   40   1
10584   2880   360  56448  2352   84   84   1
10584   2080   176  63168  1568   56   84   1
```

and its determinant is

$$\mathbf{14739701760 = 2^{17} \cdot 3^{3} \cdot 5 \cdot 7^{2} \cdot 17 \neq 0,}$$

computed by fraction-free Bareiss elimination over $\mathbb{Z}$ and, independently, by Gauss–Jordan elimination over $\mathbb{Q}$, with identical results; both routines are included in Appendix A. Hence rank $\Theta \ge 8$, and therefore rank $\Theta = 8$. Finally $\gcd(|w_1|, \ldots, |w_9|) = 1$, so w is the primitive integral generator of the relation space and is determined up to sign. ■

This minor is a self-contained certificate. A reader who wishes to check step (ii) independently need evaluate only one 8×8 integer determinant; no part of the implementation has to be rerun.

Remark (external verification of Θ). The certificate above establishes that the *displayed* matrix has rank 8. It remains to be shown that Θ is the coefficient matrix of the nine polynomials, in a form the reader may check without rerunning the computation that produced it. Six of its entries, in every row, admit closed forms that are proved by hand.

Write $A_4^{(i)}$ for the number of weight-four codewords of the i-th code. The entry of Θ at the exponent pair (a, b) counts the subsets A with $\rho(E) - \rho(A) = a$ and $|A| - \rho(A) = b$, so $|A| = 12 - a + b$.

(i) Θ_{i,(12,0)} = 1, the empty set being the only subset with $|A| = \rho(A) = 0$.

(ii) The entries of row i sum to $2^{24} = 16777216$, since every subset is counted exactly once.

(iii) Θ_{i,(0,9)} = C(24,21) = 2024. Here $|A| = 21$, and $\rho(A) < 12$ would require a nonzero codeword supported in the complementary 3-set, which is impossible because the minimum weight is at least four. Hence every one of the C(24,21) subsets of size 21 contributes, for every code.

(iv) Θ_{i,(0,8)} = C(24,20) − $A_4^{(i)}$ = 10626 − $A_4^{(i)}$. Here $|A| = 20$ and the complement is a 4-set; $\rho(A) < 12$ holds precisely when that 4-set is the support of a weight-four codeword.

(v) $\Theta_{i,(1,8)} = 20 \cdot A_4^{(i)}$. Here $|A| = 19$ and the complement is a 5-set; $\rho(A) = 11$ holds precisely when exactly one nonzero codeword is supported in that 5-set, necessarily of weight four. Each weight-four support lies in exactly 20 such 5-sets, and no 5-set contains two distinct weight-four codewords, since their sum would then have weight at most two.

(vi) $\Theta_{i,(0,7)} = C(24,19) - 20 \cdot A_4^{(i)} = 42504 - 20 \cdot A_4^{(i)}$, the complementary count to (v) among the subsets of size 19.

All six hold for all nine codes. Identity (iv) gives the entire first column of the minor above, so one of the eight columns of the rank certificate is established by hand; and (iv), (v), (vi) are finer than the weight-enumerator constraint of observation (b), pinning down individual entries rather than a combination of rows.

Theorem 7 (closed form of the relation). *Order the nine codes by A_4 as in Table 1, and let $v_1 < \dots < v_8$ be the eight distinct values 0, 6, 12, 18, 24, 30, 42, 66. Combining the two codes with $A_4 = 42$ into a single entry, the relation vector becomes*

$u = (576, -2772, 4928, -3465, 0, 924, -198, 7),$

and u is exactly the order-six divided difference on the seven values other than 24: up to a common factor,

$u_i = 1 / \prod$ over $j \neq i, v_j \neq 24$, of $(v_i - v_j)$, with $u = 0$ at $v = 24$.

Equivalently, w annihilates the first six moments of A_4 and no more:

$\Sigma\, w_i \cdot A_4^{(i)\wedge} r = 0$ for $r = 0, 1, 2, 3, 4, 5$, $\Sigma\, w_i \cdot A_4^{(i)6} = 62078607360 \neq 0$.

Moreover the two codes with $A_4 = 42$ are split by

$3 \cdot w_7 + w_8 = 0$.

Proof. The seven scaled divided differences are 576, −2772, 4928, −3465, 924, −198 and 7, agreeing with u entry by entry; a divided difference of order six annihilates every polynomial of degree at most five, which gives the six vanishing moments, and the seventh moment is evaluated directly. The final identity is $3 \cdot 99 - 297 = 0$. ■

The vanishing coefficient of $d_{10} \oplus e_7^{\wedge} \oplus 2$ is therefore not an isolated accident: that code is the one whose A_4 is the excluded node of the divided difference. The coefficient 3 in the last identity is the number of connected components of the matroid of $e_8^{\wedge} \oplus 3$, which is three, against two for $d_{16} \oplus e_8$; these are the only two of the nine whose matroids decompose, as recorded in Section 9.

What follows is not claimed. The relation is not a consequence of the coefficients of Θ being polynomial in A_4 of degree at most five: of the 76 columns only 27 lie in the eight-dimensional space spanned by the powers $A_4^{\wedge} r$ with $r \leq 5$ together with the two coordinate functionals of the codes with $A_4 = 42$. The moment vanishing is a property of the relation, not an explanation of it, and Open Problem 2 asks for the explanation.

Three observations follow. None of them is used in any proof below.

(a) The coefficient of $d_{10} \oplus e_7$^⊕2 vanishes. The unique representative whose component decomposition involves e_7 is precisely the one absent from the relation. We have no explanation for this.

(b) Two consistency constraints hold, and are forced. The coefficients sum to zero, and so does the sum weighted by A_4:

Σ w_i = 0, Σ w_i·A_4^(i) = 0.

These are not accidental. By Greene's theorem [4] the genus-one Tutte polynomial determines the weight enumerator, and by Gleason's theorem the weight enumerator of a Type II code of length 24 is determined by A_4, so the nine weight enumerators lie in a two-dimensional space spanned by the constant function and A_4. Any relation among the polynomials must annihilate that space. The two identities are therefore an independent check on the computation, and both hold exactly.

(c) Arithmetic structure. These factorisations are explained by Theorem 7. Among the eight nonzero coefficients, four are multiples of 77 = 7·11, namely 4928 = 64·77, 3465 = 45·77, 2772 = 36·77 and 924 = 12·77; the other four are not.

6. A Greene-type specialisation at genus g

This section is independent of Sections 3 to 5 and may be read on its own. Its result enters the main argument only through Corollary 12 below, which explains why the specialisation studied here cannot by itself settle the question of Section 7.

Two remarks in [1] frame this section. First, their Theorem 5.1 evaluates the ordinary Tutte polynomial as a special value of $T^{(2)}$, and their Corollary 5.2 combines this with Greene's theorem to recover the ordinary weight enumerator of a code from $T^{(2)}$; so a relation in the direction genus two to genus one is already available. Second, and in the other direction, they write that they have no relation between the complete weight enumerator of genus g and the Tutte polynomial of genus g. We establish an explicit relation on a subvariety, at the cost of passing from genus g to genus g^2. Throughout this section C is self-dual, so that Lemma 4 applies. Double evenness is not required.

6.1. The identity at genus two

By Lemma 4, each factor of $R^{(2)}$ rearranges as

u^(ρ(E)−ρ(A)) · v^(|A|−ρ(A)) = u^(k−|A|+d(A)) · v^(d(A)) = u^k · u^(−|A|) · (u·v)^(d(A)).

The choice t_a = 2 is forced if t_a is required to be a constant base, independent of A, satisfying t_a^(d(A)) = N(A) = 2^(d(A)) for every attainable value of d(A). Restricting to the subvariety t_a = 2 for all four indices,

$R^{(2)}$|(t_a = 2) = ($u_1u_2u_{\cap}u_{\cup}$)^k · Σ over (A_1,A_2) of u_1^(−|A_1|) u_2^(−|A_2|) $u_{\cap}$^(−|$A_1 \cap A_2$|) $u_{\cup}$^(−|$A_1 \cup A_2$|) · N(A_1)·N(A_2)·N($A_1 \cap A_2$)·N($A_1 \cup A_2$).

Expanding the four counting factors replaces their product by the number of quadruples (a, b, p, q) $\in C^4$ with supp(a) $\subseteq A_1$, supp(b) $\subseteq A_2$, supp(p) $\subseteq A_1 \cap A_2$ and supp(q) $\subseteq A_1 \cup A_2$. Exchange the order of summation and fix a quadruple, assigning to each coordinate its state, the pair of Booleans recording membership in A_1 and A_2. The containment constraints are coordinate-wise and the monomial is a product over coordinates, so the inner sum factorises. Write $\alpha = u_1^{-1}$, $\beta = u_2^{-1}$, $\gamma = u_\cap^{-1}$, $\delta = u_\cup^{-1}$ and $\Pi = \alpha\beta\gamma\delta$. Coordinate i in state 00 contributes 1, in state 10 it contributes $\alpha\delta$, in state 01 it contributes $\beta\delta$, and in state 11 it contributes Π. The constraints read: a_i = 1 or p_i = 1 forces $i \in A_1$; b_i = 1 or p_i = 1 forces $i \in A_2$; q_i = 1 forbids state 00. Summing the weights of the permitted states gives the following table, whose five rows exhaust the sixteen Boolean values of (a_i, b_i, p_i, q_i).

condition on (a_i, b_i, p_i, q_i)	permitted states	W(a_i, b_i, p_i, q_i)
p_i = 1, or a_i = b_i = 1	11	Π
p_i = 0, a_i = 1, b_i = 0	10, 11	$\alpha\delta + \Pi$
p_i = 0, a_i = 0, b_i = 1	01, 11	$\beta\delta + \Pi$
p_i = 0, a_i = b_i = 0, q_i = 1	10, 01, 11	$\alpha\delta + \beta\delta + \Pi$
p_i = 0, a_i = b_i = q_i = 0	00, 10, 01, 11	$1 + \alpha\delta + \beta\delta + \Pi$

Theorem 8 (Greene-type specialisation at genus two). *Let C be a self-dual binary [n,k] code. On the subvariety t_a = 2 for a ∈{1, 2, ∩, ∪},*

$$\mathbf{R^{(2)}(M(C)) = (u_1 u_2 u_\cap u_\cup)^k \cdot cwe_C^{(4)}(x_s = W(s) : s \in F_2^4),}$$

an equality of functions on that subvariety, not merely an existence statement.

For g = 1 the same argument involves a single set and a single codeword and recovers the self-dual binary specialisation of Greene's identity; Greene's theorem itself is more general, applying to arbitrary linear codes over arbitrary finite fields.

Implementation check, not a proof of Theorem 8. Both sides were computed in exact rational arithmetic, the left by enumeration of all $4^8 = 65536$ pairs of subsets and the right by all $16^4 = 65536$ quadruples of codewords. Agreement was exact for e_8 at $(u_1,u_2,u_\cap,u_\cup) = (3/2, 5/3, 7/4, 9/5)$ and $(11/7, 13/6, 5/2, 8/3)$, and for $i_2 \hat{} \oplus 4$, which is self-dual but not doubly even; the latter confirms that double evenness plays no role.

6.2. The general form

Theorem 9. *As an identity of Laurent-polynomial functions on the subvariety t_a = 2 for every index a, the genus-g Whitney rank generating function of a self-dual binary code equals an explicit specialisation of its complete weight enumerator of genus g^2.*

Proof. By Definition 1 the polynomial records g singletons, C(g,2) intersections and C(g,2) unions, that is g^2 sets, and by the rearrangement above each contributes one factor N(·) after the restriction. Expanding all g^2 counting factors yields a sum over g^2-tuples of codewords, indexed by the g^2 sets. For a fixed tuple, each coordinate i is assigned its state in {0,1}^g, recording membership of i in each A_j; the intersections and unions are determined by that state, so the containment constraints are again coordinate-wise, and

the monomial is again a product over coordinates. The inner sum therefore factorises, and the coordinate factor is a function on $F_2^{(g^2)}$, evaluated at the g^2-tuple of bits occurring at that coordinate. Summing over the g^2-tuples of codewords is by Definition 3 the complete weight enumerator of genus g^2. ■

Thus genus one corresponds to genus one, genus two to genus four, genus three to genus nine. This is a partial answer to the question raised in Remark 5.5 of [1]: a natural relation exists, but only after passing from genus g on the Tutte side to genus g^2 on the weight-enumerator side.

6.3. Full recovery by interpolation

Theorem 9 concerns a single subvariety. The following observation shows that the family of such subvarieties recovers the whole polynomial, and it identifies the price in genus.

Fix the parameters $\xi = u_1^{-1}$, $\eta = u_2^{-1}$, $\zeta = u_\cap^{-1}$, $\tau = u_\cup^{-1}$ and write

$$\Phi_C = R^{(2)} / (u_1 u_2 u_\cap u_\cup)^k = \Sigma \text{ over } (A_1, A_2) \text{ of } \xi^{|A_1|}\, \eta^{|A_2|}\, \zeta^{|A_1 \cap A_2|}\, \tau^{|A_1 \cup A_2|} \cdot t_1^{d(A_1)}\, t_2^{d(A_2)}\, t_\cap^{d(A_1 \cap A_2)}\, t_\cup^{d(A_1 \cup A_2)}.$$

Since $d(A) = \dim(C \cap F_2^A) \le k$ for every A, the degree of Φ_C in each t_a is at most k.

Lemma. *Let $m = (m_1, m_2, m_\cap, m_\cup)$ be non-negative integers and put $|m| = m_1 + m_2 + m_\cap + m_\cup$. On the subvariety $t_a = 2^{(m_a)}$,*

$$\Phi_C = \mathrm{cwe}_C^{(|m|)}\big(x_s = W(\sigma_1(s), \sigma_2(s), \sigma_\cap(s), \sigma_\cup(s)) \big),$$

where the $|m|$ coordinates of $s \in F_2^{|m|}$ are grouped into four consecutive blocks of sizes $m_1, m_2, m_\cap, m_\cup$, and $\sigma_a(s)$ denotes the logical OR of the bits of the a-th block. Here W is the function of Theorem 8, unchanged.

Proof. On that subvariety, $2^{m_a \cdot d(A)} = N(A)^{m_a}$ counts m_a-tuples of codewords supported in A. Expanding all four factors gives a sum over $|m|$-tuples of codewords. Exchanging the order of summation and fixing such a tuple, the containment constraint at coordinate i requires $i \in A_1$ if any codeword of the first or third block has a one there, and similarly for the others; that is, the constraint depends on the tuple only through the four block-ORs at that coordinate. The monomial is again a product over coordinates, so the inner sum factorises exactly as in Section 6.1, with the four arguments of W replaced by the block-ORs. A block of size zero contributes the argument 0. ■

For $j = 0, \dots, k$ let $\ell_j(t) = \prod$ over $i \ne j$ of $(t - 2^i)/(2^j - 2^i)$ be the Lagrange basis on the nodes $2^0, \dots, 2^k$. Because Φ_C has degree at most k in each t_a, interpolation on the product grid is exact, and the interpolation coefficients depend only on k, not on the code.

Proposition 10. *As an identity in $\mathbb{Q}[\xi, \eta, \zeta, \tau][t_1, t_2, t_\cap, t_\cup]$,*

$$\Phi_C = \Sigma \text{ over } m \in \{0,\dots,k\}^4 \text{ of } \ell_{m_1}(t_1) \cdot \ell_{m_2}(t_2) \cdot \ell_{m_\cap}(t_\cap) \cdot \ell_{m_\cup}(t_\cup) \cdot \mathrm{cwe}_C^{(|m|)}\big(x_s = W(\sigma_1(s), \sigma_2(s), \sigma_\cap(s), \sigma_\cup(s)) \big).$$

In particular the genus-two Whitney rank generating function of a self-dual binary [n,k] code is determined by, and explicitly reconstructible from, its complete weight enumerators of genus at most 4k; Proposition 11 below sharpens this to k. At genus g the same argument gives the bound k·g², one factor k for each of the g² sets recorded by Definition 1.

Proposition 11 (sharpened bound). *Put*

$e_0 = d(A_1 \cap A_2)$, $e_1 = d(A_1) - d(A_1 \cap A_2)$, $e_2 = d(A_2) - d(A_1 \cap A_2)$, $e_3 = d(A_1 \cup A_2) - d(A_1) - d(A_2) + d(A_1 \cap A_2)$.

All four are non-negative, the first three because $A_1 \cap A_2$ is contained in A_1 and in A_2, and the fourth by modularity of the dimension function. Writing $s_0 = t_1 t_2 t_\cap t_\cup$, $s_1 = t_1 t_\cup$, $s_2 = t_2 t_\cup$ and $s_3 = t_\cup$, the monomial in the t variables becomes $s_0^{(e_0)} s_1(e_1) s_2^{(e_2)} s_3(e_3)$, and

$e_0 + e_1 + e_2 + e_3 = d(A_1 \cup A_2) \le k$.

Thus Φ_C has total degree at most k in the variables s, rather than degree k in each of four variables. At the point t_a = 2^(m_a) one has s_i = 2^(n_i) with $n_3 = m_\cup$, $n_1 = m_1 + m_\cup$, $n_2 = m_2 + m_\cup$ and $n_0 = m_1 + m_2 + m_\cap + m_\cup = |m|$, so the genus spent at a node is exactly the exponent n_0. The nodes with $|m| \le k$ are in bijection with the monomials of total degree at most k, both being counted by C(k+4,4), and the resulting evaluation matrix V, with entries V_{m,e} = 2^(⟨n(m), e⟩), is nonsingular. Hence the genus-two Whitney rank generating function of a self-dual binary [n,k] code is determined by, and reconstructible from, its complete weight enumerators of genus at most k.

Proof. Non-negativity of e_0, e_1, e_2 is the containment of $A_1 \cap A_2$ in A_1 and A_2; non-negativity of e_3 is the modular inequality $d(A_1) + d(A_2) \le d(A_1 \cap A_2) + d(A_1 \cup A_2)$, which holds because the codewords supported in A_1 and those supported in A_2 generate a subspace of the codewords supported in $A_1 \cup A_2$ and intersect in those supported in $A_1 \cap A_2$. The substitution is immediate, and the sum of the four exponents telescopes to $d(A_1 \cup A_2)$, which is at most k. The stated bijection sends a node m to the monomial exponent vector of the same total degree, and both sets are the lattice points of a simplex of side k in four variables. ■

The bound k replaces the bound 4k of Proposition 10, an improvement by the factor $g^2 = 4$. At length 24, where k = 12, the reconstruction genus falls from 48 to 12. Propositions 10 and 11 are reconstruction statements about a single code; they are not used in the proof of Theorem 18 below, which relies only on the specialisation of Lemma 13 below and the finite-field certificate of Section 7.3. Nonsingularity of V is certified as follows: for every k from 1 to 12 the rank of V modulo the prime $2^{61} - 1$ equals its size, namely 5, 15, 35, 70, 126, 210, 330, 495, 715, 1001, 1365 and 1820, and a matrix of full rank modulo a prime is nonsingular over $\mathbb{Q}$. The case k = 12 is the one used at length 24. The change of variables, the total degree and the node count were verified in exact rational arithmetic for k up to 4, where the reconstruction was also carried out and compared with the direct value at a point off the grid.

Verification. Both sides were compared in exact rational arithmetic for two codes of length four, at three parameter tuples including negative and fractional values, and at two points off the interpolation grid; all twelve comparisons agreed. Using the same interpolation

coefficients for two different codes confirms that they are code-independent, as the statement requires. The bound k is attained: the exponent k occurs for both e_8 and i_2^⊕2.

Relation to earlier work. Proposition 10 must be placed carefully. For the ordinary Tutte polynomial the corresponding statement is due to Britz [7], who proved that the set of higher weight enumerators of a linear code over a finite field is equivalent to the Tutte polynomial of the code, and gave an explicit expression for the Tutte polynomial in terms of subcode weights; he also proved generalisations for codeword m-tuples. The case g = 1 of Proposition 10 is therefore not new. What is added here is the extension to the genus-g Tutte polynomial of [1, 2], where the factor g^2 arises because that polynomial records the pairwise intersections and unions in addition to the g sets themselves. Together with Theorem 9, this answers the question raised at the end of Section 5.2 of [1]: a relation does exist, but between genus g on the Tutte side and genus at most $k \cdot g^2$ on the weight-enumerator side.

6.4. Related work

Greene's theorem has been generalised in several directions. Britz and Shiromoto [6] proved a MacWilliams-type identity for matroids and used it to generalise Greene's theorem with respect to m-tuples of codewords; Britz [7] described further connections between weight enumerators and Tutte polynomials; Britz, Shiromoto and Westerbäck [8] extended the identity to m-tuple weight enumerators over finite Frobenius rings by way of demi-matroids; and Chakraborty, Miezaki and Oura [9] gave a harmonic analogue.

All of these pair an m-tuple weight enumerator with a rank generating function of a matroid or demi-matroid, keeping the number of codewords equal to the number of subsets. Theorem 8 differs in the object on the matroid side and in the pairing. The genus-g Tutte polynomial of [1, 2] records not only g subsets but also their pairwise intersections and unions, and it is exactly this extra structure that produces g^2 rather than g codewords. The authors of [1] state that no relation between their polynomial and the genus-g complete weight enumerator was known; Theorem 9 supplies one, at the cost of the shift $g \mapsto g^2$.

6.5. Consequence for the restricted problem

Independence of genus-g weight enumerators is monotone in g: setting c_g = 0 in Definition 3 and collapsing the variables x_(s,1) to zero and x_(s,0) to x_s recovers the genus-(g−1) enumerator, so independence at genus g−1 implies independence at genus g. Since the least genus of linear independence at length 24 is six, the genus-four and genus-five weight enumerators of the nine codes are linearly dependent. Theorem 8 gives the following.

Corollary 12. *The genus-two Whitney rank generating functions of the nine Type II codes of length 24, restricted to the subvariety t_a = 2, are linearly dependent.*

Theorem 18 below states that the unrestricted functions are linearly independent. There is no tension: restriction to a subvariety can only enlarge the relation space. Precisely, any linear independence present in the full genus-two polynomials is necessarily lost under

this specialisation. The same phenomenon occurs at length 16, where the two codes have identical weight enumerators while their genus-one Tutte polynomials differ.

7. Genus two suffices

We now prove the main theorem. Step (i) of the architecture occupies Section 7.1, the algorithm required for step (iii) occupies Section 7.2, and the certificate and the theorem occupy Section 7.3.

7.1. Reduction to a single scalar

Lemma 13. *$Rel(T^{(2)}) \subseteq Rel(T^{(1)})$ for any finite family of matroids on a common ground set E.*

An equivalent identity, in the Tutte variables and in the direction from genus g + 1 to genus g, is stated in [1] immediately after their Definition 1.3. We give a direct proof of the form we need.

Proof. Setting u_a = v_a = 1 for a ∈ {2, ∩, ∪} makes the corresponding factors identically one, so the summand no longer depends on A_2 and the free sum over A_2 contributes a factor 2^|E|:

$$\mathbf{R^{(2)}|(u_2 = v_2 = u_\cap = v_\cap = u_\cup = v_\cup = 1) = 2\text{^}|E| \cdot R^{(1)}(u_1, v_1).}$$

A vanishing linear combination of the genus-two functions therefore specialises to a vanishing linear combination of the genus-one functions with the same coefficients. This specialisation is applied to the Whitney rank generating functions, so no substitution u ↦ x−1 is involved at this step; Lemma 2 then transfers the inclusion to the Tutte polynomials. ■

Combining Lemma 13 with Theorem 6, $Rel(T^{(2)}) \subseteq \langle w \rangle$. The genus-two question therefore reduces to a single scalar: whether $\Sigma\, w_i \cdot T_i^{(2)}$ vanishes identically.

7.2. An O(n·2ⁿ) exact finite-field evaluation

By Section 7.1, a single exact evaluation of the scalar $\Sigma\, w_i \cdot R_i^{(2)}$ at one point settles the question. A direct evaluation from Definition 1 is out of reach, since it ranges over 4^n ordered pairs of subsets. The following reduction removes that obstacle.

Using Lemma 4 and $|A_1 \cup A_2| = |A_1| + |A_2| - |A_1 \cap A_2|$, and writing t_a = u_a·v_a,

$$\mathbf{R^{(2)} / (u_1u_2u_\cap u_\cup)\text{^}k = \Sigma \text{ over } (A_1,A_2) \text{ of } \xi\text{^}|A_1| \cdot \eta\text{^}|A_2| \cdot \zeta\text{^}|A_1 \cap A_2| \cdot t_1\text{^}d(A_1) \cdot t_2\text{^}d(A_2) \cdot t_\cap\text{^}d(A_1 \cap A_2) \cdot t_\cup\text{^}d(A_1 \cup A_2),}$$

with $\xi = (u_1 u_\cup)^{-1}$, $\eta = (u_2 u_\cup)^{-1}$ and $\zeta = u_\cup \cdot u_\cap^{-1}$.

Impose t_∪ = 1, that is $v_\cup = u_\cup^{-1}$ with u_∪ arbitrary and nonzero. This single condition defines a seven-dimensional subvariety of the eight-variable space. After removing the common monomial prefactor, the normalised expression depends only on the six combinations ξ, η, ζ, t_1, t_2, t_∩.

Lemma 14. *Every tuple (ξ, η, ζ, t_1, t_2, t_∩) of nonzero field elements arises from an actual point of the subvariety.*

Proof. Take u_∪ = 1, $u_1 = \xi^{-1}$, $u_2 = \eta^{-1}$, u_∩ = ζ^{-1}, then $v_1 = t_1 \cdot u_1^{-1}$, $v_2 = t_2 \cdot u_2^{-1}$, v_∩ = $t_\cap \cdot u_\cap^{-1}$ and v_∪ = 1. ■

With t_∪ = 1 the last factor disappears and the summand becomes $f_1(A_1) \cdot f_2(A_2) \cdot h(A_1 \cap A_2)$, where

$f_1(A) = \xi^{|A| \cdot t_1} d(A)$, $f_2(A) = \eta^{|A| \cdot t_2} d(A)$, $h(I) = \zeta^{|I| \cdot t_\cap} d(I)$.

Let $\hat{h}$ be the subset Möbius transform of h, so that h(I) = Σ over S ⊆ I of $\hat{h}(S)$. Substituting and exchanging the order of summation, a pair (A_1, A_2) contributes to the term indexed by S exactly when $S \subseteq A_1 \cap A_2$, that is, exactly when $S \subseteq A_1$ and $S \subseteq A_2$. Hence

Σ over (A_1, A_2) of $f_1(A_1) \cdot f_2(A_2) \cdot h(A_1 \cap A_2)$ = Σ over $S \subseteq E$ of $\hat{h}(S)$ · (Σ over $A \supseteq S$ of $f_1(A)$) · (Σ over $A \supseteq S$ of $f_2(A)$).

The two inner sums are superset-sum transforms of f_1 and f_2.

Proposition 15. *On the subvariety t_∪ = 1, the genus-two Whitney rank generating function of a self-dual binary code is evaluated exactly by one subset Möbius transform, two superset-sum transforms and one inner product, using 3n·2^(n−1) additions or subtractions together with $O(2^n)$ multiplications, hence $O(n \cdot 2^n)$ arithmetic operations.*

At n = 24 the three transforms perform exactly $3 \cdot 24 \cdot 2^{23} \approx 6.04 \times 10^8$ pairwise array updates; the pointwise multiplications and modular reductions contribute a further $O(2^n)$ operations. A direct evaluation from Definition 1 would instead enumerate the $4^{24} \approx 2.81 \times 10^{14}$ ordered pairs of subsets.

Exactness. All arithmetic is carried out modulo the Mersenne prime $p = 2^{31} - 1 = 2147483647$ in 64-bit signed integers. Every modular product is bounded by $(p-1)^2 \approx 4.61 \times 10^{18} < 2^{63} - 1 \approx 9.22 \times 10^{18}$, and the inner product is accumulated in blocks of 2^{21} entries, each block sum bounded by $2^{21} \cdot (p-1) \approx 4.5 \times 10^{15}$. The block size is chosen so that the unreduced 64-bit accumulation cannot overflow.

Validation. The transform was validated against direct enumeration of all 4^8 pairs of subsets for e_8 and for i_2^⊕4, and the reduced form in (ξ, η, ζ, t_1, t_2, t_∩) was validated against Definition 1 in terms of ρ; all comparisons were exact in rational arithmetic.

7.3. The certificate and the theorem

Let $R_i^{(2)}$ denote the genus-two Whitney rank generating function of the i-th code in the ordering of Table 1, and let

Φ := Σ (i = 1 to 9) $w_i \cdot R_i^{(2)} / (u_1 u_2 u_\cap u_\cup)^k$.

All nine codes have the same dimension k = 12, so the prefactor is common to all nine terms and is a nonzero field element at every point considered; consequently Φ vanishes identically if and only if Σ $w_i \cdot R_i^{(2)}$ does.

Definition 16 (three-part certificate). *Let $F_1, \dots, F_m$ be polynomials with integer coefficients. A three-part certificate for the linear independence of $F_1, \dots, F_m$ consists of three items:*

(a) a specialisation $F_i \mapsto G_i$, applied identically to all m polynomials, together with the coefficient matrix Θ of $G_1, \dots, G_m$ in a stated ordered monomial basis, and a vector $w \in \mathbb{Z}^m$ with $\Theta^T w = 0$. The matrix Θ is printed in full in Appendix C, and is also generated deterministically from the printed generator matrices by the reference implementation. Since a linear relation among the F_i specialises to the same relation among the G_i, these data give $\mathrm{Rel}(F) \subseteq \mathrm{Rel}(G)$ and $\mathrm{Rel}(G) \subseteq \langle w \rangle$ once (b) is supplied;

(b) an integer minor of Θ with nonvanishing determinant, whose order equals $m - 1$, establishing $\operatorname{rank} \Theta = m - 1$ and hence $\dim \mathrm{Rel}(G) = 1$;

(c) an explicit integer point together with a prime p, at which the value of $\Sigma\, w_i F_i$, reduced modulo p, is nonzero.

The three items are exactly the data displayed in this paper: the specialisation of Lemma 13, with the matrix Θ, the vector w and the identity $\Theta^{\mathrm{T}} w = 0$ of Theorem 6; the 8×8 minor of Theorem 6 with determinant 14739701760; and the point P_1 with the residue of Lemma 17 below. Items (a) and (b) are verified by hand arithmetic on displayed integers; item (c) requires one evaluation of an explicit polynomial at an explicit integer point. No part of the implementation need be rerun.

The three items are independent of how they were found: a reader who distrusts the search of Section 4, the transforms of Sections 3 and 7.2, or the implementation may disregard them and check the certificate directly.

Lemma 17 (computational certificate). *At the point $P_1 = (\xi, \eta, \zeta, t_1, t_2, t_\cap) = (2, 3, 5, 7, 11, 13)$,*

$\Phi(P_1) \equiv 749926473 \pmod{2147483647}$.

Proof. Direct exact finite-field evaluation by Proposition 15, applied to the generator matrices printed in Appendix B. ■

The finite-field computation serves only as a one-sided nonvanishing certificate; a zero residue would not be used to infer a zero value.

Two further points were evaluated as independent checks against implementation error; only Lemma 17 is used in the proof.

point $(\xi, \eta, \zeta, t_1, t_2, t_\cap)$	residue mod $2^{31}-1$
(2, 3, 5, 7, 11, 13)	749926473
(3, 5, 7, 11, 13, 17)	1241429917
(5, 7, 11, 13, 17, 19)	1907627176

The points are fixed small integers, specified independently of the reported output and listed here in full; no random sampling is used, and the argument does not invoke the

Schwartz–Zippel lemma. No randomness enters the proof once the nine representatives printed in Appendix B are fixed; the randomised search of Section 4 is a construction method only. In the six normalised variables ξ, η, ζ, t_1, t_2, $t_\cap$ every exponent occurring in Φ is nonnegative, so Φ is an ordinary polynomial with integer coefficients. A single point with nonzero value therefore suffices, since it exhibits a point at which Φ does not vanish.

Together with Theorem 6 and Lemma 13, this residue completes the argument.

Certificate summary.

item	value
prime	$p = 2^{31} - 1 = 2147483647$
point $P_1 = (\xi, \eta, \zeta, t_1, t_2, t_\cap)$	(2, 3, 5, 7, 11, 13)
residue $\Phi(P_1) \bmod p$	749926473
code ordering	Table 1
input data	Appendix B

Definition. For a finite family of codes, the least Tutte-separating genus is the smallest g for which the associated genus-g Tutte polynomials are linearly independent over $\mathbb{Q}$.

Theorem 18. *The genus-two Tutte polynomials of representatives of the nine equivalence classes of Type II binary self-dual codes of length 24 are linearly independent. Consequently the least genus at which the genus-g Tutte polynomials of the Type II codes of length 24 become linearly independent is exactly two.*

Proof. By Lemma 2 the linear relations among the nine genus-two Tutte polynomials are exactly those among the corresponding Whitney rank generating functions, so we work with the normalised $R^{(2)}$ of Section 7.2. A full 9×9 determinant is unnecessary: Theorem 6 reduces the possible relation space to the one-dimensional $\langle w \rangle$, so one nonvanishing specialisation of w suffices. If w were a relation, its value at every point would vanish as a rational number, hence modulo every prime not dividing the denominators; the computed residue is nonzero, so w is not a relation. The three items of Definition 16 are supplied by Theorem 6 and Lemma 17. By Lemma 14, P_1 is realised by a point of the subvariety $t_\cup = 1$ with nonzero coordinates, namely $(u_1, u_2, u_\cap, u_\cup) = (1/2, 1/3, 1/5, 1)$, at which the prefactor $(u_1u_2u_\cap u_\cup)^k$ is nonzero. At that integer point every term of the normalised sum is an integer, so $\Phi(P_1) \in \mathbb{Z}$ and the reported residue is its reduction modulo p. By Lemma 17 that residue is nonzero, so Φ does not vanish at P_1 and therefore does not vanish identically. Since the prefactor is nonzero, $\Sigma\, w_i \cdot R_i^{(2)}$ does not vanish identically, and by Lemma 2 neither does $\Sigma\, w_i \cdot T_i^{(2)}$. Hence $w \notin \mathrm{Rel}(T^{(2)})$. By Lemma 13 and Theorem 6, $\mathrm{Rel}(T^{(2)}) \subseteq \langle w \rangle$, so $\mathrm{Rel}(T^{(2)}) = 0$ and the nine polynomials are linearly independent. Linear dependence at genus one is Theorem 6 together with the theorem of Miezaki et al. ■

A zero residue would have been inconclusive, since a nonzero integer may be divisible by p; the certificate is one-sided in the safe direction.

7.4. The observed pattern

Writing g_T(n) for the Tutte value and g_W(n) for the Witt value, the two known pairs are

g_T(2) = 1 = 3/3 = g_W(2)/3, g_T(3) = 2 = 6/3 = g_W(3)/3.

Whether this persists is open. Two data points are not evidence for a general law, and we record the coincidence only because the case m = 4 lies within reach of the methods of this paper and would decide it.

7.5. The full eight-variable polynomial

Proposition 15 evaluates the genus-two function only on the subvariety $t_\cup = 1$, which is all that Theorem 18 requires. We record here that the restriction can be removed at a controlled cost.

Proposition 19. *The full eight-variable genus-two Whitney rank generating function of a binary linear code of length n is evaluated exactly by two subset Möbius transforms followed by a sum over ordered pairs of disjoint subsets, hence in $O(3^n)$ arithmetic operations, in place of the 4^n ordered pairs of subsets required by Definition 1.*

Proof. Since $A_1 \cup A_2$ is the complement of $A_1^{\,c} \cap A_2^{\,c}$, the factor attached to the union is an intersection factor in the complementary variables. Write

$$\mathbf{h(I) = \zeta^{|I|\cdot t_\cap} d(I),\ k(J) = \tau^{(n-|J|)\cdot t_\cup}(d(J^c)),}$$

so that the summand of Definition 1 is $f_1(A_1)\cdot f_2(A_2)\cdot h(A_1 \cap A_2)\cdot k(A_1^{\,c} \cap A_2^{\,c})$ with f_1 and f_2 as in Section 7.2. Let $\hat{h}$ and $\hat{k}$ be the subset Möbius transforms of h and k, so that h(I) is the sum of $\hat{h}(S)$ over $S \subseteq I$ and k(J) is the sum of $\hat{k}(S')$ over $S' \subseteq J$. Substituting and exchanging the order of summation, a pair (A_1, A_2) contributes to the term indexed by (S, S') exactly when $S \subseteq A_1 \cap A_2$ and $S' \subseteq A_1^{\,c} \cap A_2^{\,c}$, that is, exactly when S is contained in both sets and S' meets neither. In particular S and S' are disjoint, and

$$\mathbf{\Sigma \text{ over } (A_1,A_2) = \Sigma \text{ over disjoint } (S,S') \text{ of } \hat{h}(S)\cdot\hat{k}(S')\cdot F_1(S,S')\cdot F_2(S,S'),}$$

where $F_i(S,S')$ is the sum of $f_i(A)$ over the A containing S and disjoint from S'. The two transforms cost $n\cdot 2^{(n-1)}$ operations each, and the ordered pairs of disjoint subsets number 3^n. ■

The identity was verified in exact rational arithmetic at n = 8 for e_8, where both routes give 649580308340972973589002406369604 0.

n	4^n, from Definition 1	3^n, Proposition 19	2^n, on the subvariety
24	2.8×10^{14}	2.8×10^{11}	1.7×10^{7}
32	1.8×10^{19}	1.9×10^{15}	4.3×10^{9}

The exponent 3 rather than 2 has a structural reason. Intersection is closed downwards in the subset lattice and union upwards, so the two coupling factors call for Möbius inversion in opposite directions; imposing both at once replaces the Boolean lattice of subsets by the

lattice of ordered pairs of disjoint subsets, whose size is 3^n. We do not know whether some other transform achieves $O(n \cdot 2^n)$ for the unrestricted polynomial.

8. The separation lattice

Sections 6 and 7 establish what the genus-two polynomial can do at length 24, but not how much of a weight enumerator it resolves. This section answers that, in a form independent of the length and of the family.

The observation of this section is that the local pattern space collapses in two mathematically distinct stages,

$2^{(g^2)} \longrightarrow |\mathcal{L}_g| \longrightarrow 2^g$,

the first combinatorial and the second algebraic. A pattern is seen only through its admissible state set, which produces the separation lattice $\mathcal{L}_g$; the resulting local weights then satisfy linear relations, reducing the effective dimension from $|\mathcal{L}_g|$ to 2^g. At genus six this reads 68 719 476 736 ⟶ 40 069 ⟶ 64.

Write Λ_g for the index set of Definition 1: the g singletons, the C(g,2) pairwise intersections and the C(g,2) pairwise unions, so that $|\Lambda_g| = g^2$. To each $\lambda \in \Lambda_g$ attach the set $A_\lambda \subseteq \mathbb{F}_2^g$ of admissible states, namely $\{a : a_i = 1\}$ for a singleton, $\{a : a_i = a_j = 1\}$ for an intersection and $\{a : a_i \vee a_j = 1\}$ for a union. Here the state of a coordinate is the vector recording its membership in each A_j, and the weight of a state is the product of the variables attached to those λ for which the state lies in A_λ; the zero state has weight one.

8.1. The local weight and its level sets

Theorem 20. *Let s be the pattern of a g^2-tuple of codewords at a fixed coordinate. The local weight of Theorem 9 is*

$W_g(s) = \Sigma$ over $a \in A(s)$ of $\mathrm{wt}(a)$, where $A(s) = \bigcap$ over the λ with $s_\lambda = 1$ of A_λ.

In particular $W_g(s)$ depends on s only through the subset $A(s)$ of $\mathbb{F}_2^g$.

Proof. Fix a coordinate. The containment constraint attached to λ requires that if the codeword indexed by λ has a one at this coordinate, then the coordinate lies in A_λ, that is, the state a satisfies $a \in A_\lambda$. The constraints active at this coordinate are exactly those λ with $s_\lambda = 1$, and they are imposed simultaneously, so the admissible states form the intersection of the corresponding A_λ. The local factor is the sum of the weights of the admissible states, and no other feature of s enters. ■

For g = 2 the five values of the table in Section 6.1 are recovered as the admissible sets of sizes 4, 3, 2, 2 and 1.

Proposition 21. *Let z_a, for $a \in \mathbb{F}_2^g$, be algebraically independent indeterminates and put $W_\ell = \Sigma$ over $a \in \ell$ of z_a for $\ell \in \mathcal{L}_g$. Then $\ell \mapsto W_\ell$ is injective. Consequently, for any specialisation of the state weights outside the finite union of the hyperplanes $W_\ell = W_{\ell'}$ with $\ell \neq \ell'$, distinct members of $\mathcal{L}_g$ have distinct local weights.*

Proof. Let $\ell \neq \ell'$ and choose a in the symmetric difference. The coefficient of z_a in W_ℓ − W_ℓ' is 1 or −1, so that difference is a nonzero linear form. Two distinct members therefore have distinct local weights as formal polynomials, and they coincide only on a proper algebraic subset of the space of state weights. ■

Combining Theorem 20 with Proposition 21: for algebraically independent state weights, W_g(s) = W_g(s′) if and only if A(s) = A(s′). The statement was also verified by direct enumeration at g = 2, 3 and 4, over all 16, 512 and 65536 patterns respectively.

8.2. The lattice and its size

Definition 22. The separation lattice $\mathcal{L}$_g is the family of subsets of F_2^g obtained as intersections of subfamilies of the state sets A_λ, the empty intersection being all of F_2^g.

By Theorem 20 the members of Definition 22 are precisely the values taken by A(s), and by Proposition 21 distinct members give distinct local weights for algebraically independent state weights, although those weights need not be linearly independent, as Theorem 26 below shows. The size of $\mathcal{L}$_g is therefore the number of distinct local weights available at genus g, generically.

Theorem 23. *The size of $\mathcal{L}$_g equals Σ over r from 0 to g of C(g, r) · 2^C(r,2).*

Proof. We first put an arbitrary subfamily into a canonical form. An intersection constraint on the pair {i, j} is equivalent to the two singleton constraints on i and on j, so every intersection constraint may be replaced by singleton constraints. Let T be the resulting set of forced components. A union constraint on a pair meeting T is satisfied automatically and may be discarded. What remains is the set T together with a graph G on the r = g − |T| free components, whose edges are the surviving union constraints, and the corresponding member of $\mathcal{L}$_g is

A(T, G) = { a : a_i = 1 for i ∈ T, and a_i ∨ a_j = 1 for every edge {i, j} of G }.

Every member of $\mathcal{L}$_g arises in this way, so the count is at most Σ_r C(g,r)·2^C(r,2).

The pair (T, G) is recovered from A(T, G), so distinct pairs give distinct members and the count is exact. First, T = { i : a_i = 1 for every a ∈ A }: a component outside T is left free by every surviving constraint and may be set to zero in some admissible state. Second, with T known, fix two free components i and j and let a(i,j) be the state that is one on T and on every free component except i and j, and zero at i and at j. Every union constraint on a pair other than {i, j} involves a component set to one and is satisfied, while the constraint on {i, j} fails. Hence

{i, j} ∈ E(G) if and only if a(i,j) ∉ A(T, G),

so each edge is detected separately and G is recovered. ■

Corollary 24. *The size of $\mathcal{L}$_g is the number of labelled graphs on at most g vertices.* The values agree with sequence A006896 [13]; that agreement is an independent numerical cross-check, not part of the proof of Theorem 23.

The first values, computed independently by closure of the intersection family and by the formula of Theorem 23, agree exactly, and by Corollary 24 they are the corresponding values of the graph-counting sequence.

g	size of Λ_g	patterns 2^(g²)	size of ℒ_g	first-stage retention
1	1	2	2	1
2	4	16	5	3.1×10^{-1}
3	9	512	18	3.5×10^{-2}
4	16	65 536	113	1.7×10^{-3}
5	25	33 554 432	1 450	4.3×10^{-5}
6	36	68 719 476 736	40 069	5.8×10^{-7}

The graph interpretation follows from Definition 1 rather than being a coincidence: that definition records pairwise intersections and unions and nothing of higher arity, so the residual constraints among the free components are relations between pairs, the edges of a graph. Were triples recorded as well, the count would be one of hypergraphs, and both the exponent g^2 and the closed form would change.

The five members of $\mathcal{L}_2$, with the number of patterns falling in each, are as follows.

member of $\mathcal{L}_2$	size	patterns
all of F_2^2	4	1
a_1 or a_2	3	1
$a_1 = 1$	2	2
$a_2 = 1$	2	2
$a_1 = a_2 = 1$	1	10

The collapse is markedly uneven: ten of the sixteen patterns fall together in the smallest member.

8.3. The kernel and the criterion

Write L_g for the linear map carrying a complete weight enumerator to the genus-g Whitney rank generating function, as supplied by Proposition 10, in which the genus-g function is an explicit linear combination, with code-independent coefficients, of complete weight enumerators of genus at most $k \cdot g^2$.

Let V_g be the vector space spanned by the complete-weight-enumerator monomials arising from g^2-tuples of binary states, and let P_g denote the polynomial ring $\mathbb{Q}[X_\ell : \ell \in \mathcal{L}_g]$. Define the count map Q_g from V_g to P_g by

$Q_g(m) = \prod$ over the members ℓ of the lattice of $X_\ell^{N_\ell(m)}$,

where $N_\ell(m)$ is the number of coordinates of the monomial m whose admissible set is ℓ, and the local-weight evaluation ev_g from P_g to the ring of Tutte variables, given on generators by

ev_g(X_ℓ) = W_ℓ.

Theorem 25 (two-stage factorisation). *L_g = ev_g ∘ Q_g. The kernel of the first stage is*

ker Q_g = span { m − m′ : N_ℓ(m) = N_ℓ(m′) for every member ℓ of the lattice },

and consequently ker L_g = Q_g^{−1}(ker ev_g).

Proof. By the coordinate-wise factorisation of Theorem 9 the image of a monomial under L_g is the product over coordinates of the local weights, and by Theorem 20 each local weight is determined by the member of $\mathcal{L}$_g containing that coordinate's admissible set. The image is therefore the product of the W_ℓ raised to the counts N_ℓ, which is exactly ev_g(Q_g(m)). Two monomials have the same image under Q_g precisely when their count vectors agree, which gives the stated kernel of the first stage; the last assertion is the definition of a preimage. ■

The two stages lose information for different reasons. The first loss is combinatorial, the passage s ↦ A(s); the second is algebraic, and is not vacuous. Write K_g = ker ev_g for the local-weight relation space.

The two stages are studied in turn: Theorem 25 below describes the first, Theorem 26 below the second.

Theorem 26. *The local weights span a space of dimension exactly 2^g, so K_g is the ideal generated by |$\mathcal{L}$_g| − 2^g independent linear forms in the X_ℓ. In particular K_1 = 0, while K_g ≠ 0 for every g ≥ 2.*

Proof. Write W_ℓ = Σ over a ∈ ℓ of z_a in the state weights z_a, a ∈ $\mathbb{F}_2$^g. For every T ⊆ {1, …, g} the set A_T = { a : a_i = 1 for all i ∈ T } lies in $\mathcal{L}$_g, being an intersection of singleton constraints, and the indicator vectors of the A_T are triangular with respect to inclusion of the T, hence linearly independent. They are 2^g in number, so the local weights span the whole space of linear forms in the z_a, of dimension 2^g. The map ev_g is induced by a linear map on the degree-one parts, whose image has dimension 2^g. After a linear change of coordinates in the X_ℓ it therefore becomes the quotient homomorphism $\mathbb{Q}[Y_1, \ldots, Y_{|\mathcal{L}_g|}] \to \mathbb{Q}[Y_1, \ldots, Y_{2^g}]$ sending the remaining |$\mathcal{L}$_g| − 2^g variables to zero, and the kernel of such a homomorphism is the ideal generated by those variables. Pulling back, ker ev_g is generated by |$\mathcal{L}$_g| − 2^g independent linear forms in the X_ℓ. For g = 1 the lattice has two members and $2^1 = 2$, so there is no relation; for g ≥ 2 the count is positive. ■

The first relation at genus two is inclusion and exclusion on a pair,

$W(a_1 \vee a_2) - W(a_1 = 1) - W(a_2 = 1) + W(a_1 = a_2 = 1) = 0$,

which holds identically in the state weights. The counts are:

g	size of $\mathcal{L}$_g	2^g	independent relations
1	2	2	0
2	5	4	1
3	18	8	10

g	size of $\mathcal{L}_g$	2^g	independent relations
4	113	16	97
5	1 450	32	1 418

Theorem 27 (an explicit basis for K_g). *Write $A_U = \{ a : a_i = 1 \text{ for all } i \in U \}$ for $U \subseteq \{1, \ldots, g\}$; these are the 2^g members of $\mathcal{L}_g$ with empty graph, and by Theorem 26 their local weights form a basis of the image of ev_g. For a member $A(T, G)$ with edge set $E(G)$, the indicator function satisfies*

$$\mathbf{1_{A(T,G)} = 1_{A_T} \cdot \prod_{\{i,j\} \in E(G)} \left(1_{A_{T \cup \{i\}}} + 1_{A_{T \cup \{j\}}} - 1_{A_{T \cup \{i,j\}}} \right),}$$

where the product is expanded using $1_{A_S} \cdot 1_{A_{S'}} = 1_{A_{S \cup S'}}$, so that the right-hand side is a linear combination of the 1_{A_U}. The corresponding linear forms in the X_ℓ, one for each member with non-empty graph, form a basis of K_g.

Proof. A state lies in $A(T, G)$ exactly when it lies in A_T and satisfies $a_i \vee a_j$ for every edge, so its indicator is the product of the indicators of those conditions. For a single edge, inclusion and exclusion give $1_{a_i \vee a_j} = 1_{a_i = 1} + 1_{a_j = 1} - 1_{a_i = a_j = 1}$, and intersecting with A_T turns the three terms into $A_{T \cup \{i\}}$, $A_{T \cup \{j\}}$ and $A_{T \cup \{i,j\}}$. Expanding the product over the edges and reducing every monomial by $1_{A_S} \cdot 1_{A_{S'}} = 1_{A_{S \cup S'}}$ yields a linear combination of the 1_{A_U}. Passing to local weights, each member with non-empty graph gives one linear form in K_g. These forms are independent. Each contains the variable of its own member with coefficient one, and that variable occurs in no other form: the other forms involve only their own member and the empty-graph members A_U, and by the recovery argument in the proof of Theorem 23 a member with non-empty graph is never equal to a member with empty graph. Their number is the number of members with non-empty graph, namely $|\mathcal{L}_g| - 2^g$, which by Theorem 26 is the dimension of the space of linear relations, so they form a basis. ■

The single relation at genus two is the case $T = \emptyset$ with one edge, which is the identity displayed above. The expansion was verified on every state, for every member, at $g = 2$, 3 and 4, giving 1, 10 and 97 relations respectively. Equivalently, every variable indexed by a member with non-empty graph is eliminated in favour of the empty-graph variables, so the quotient admits the canonical presentation P_g / K_g is isomorphic to $\mathbb{Q}[X_{A_U} : U \subseteq \{1, \ldots, g\}]$, a polynomial ring in 2^g variables.

Definition 28. The lattice-count enumerator of C at genus g is the polynomial in P_g obtained by summing, over all tuples of codewords, the product over the members ℓ of $\mathcal{L}_g$ of X_ℓ raised to the number of coordinates whose admissible set is ℓ. Thus the genus-g Whitney rank generating function is its image under ev_g.

Corollary 29 (necessary condition). *If the genus-g Whitney rank generating functions of a family of self-dual codes are linearly independent, then so are their lattice-count enumerators. Consequently the least g at which the lattice-count enumerators become independent is a lower bound for the least separating genus.*

Proof. A vanishing linear combination of the lattice-count enumerators maps under ev_g to the same combination of the genus-g functions. ■

The converse fails in general, since ev_g has a nonzero kernel for g ≥ 2 by Theorem 26; the lattice-count enumerator is a complete first-stage invariant, not a complete invariant for the genus-g polynomial. What the separation lattice therefore provides is a reduction of the first stage of the problem to a finite state space determined solely by the index structure, before any code is examined, together with an exact description of the residual algebraic stage.

9. The transition at length 24, structurally

Genus one. Here $\mathcal{L}_1$ has two members, all of F_2 and the set where a = 1, so a code is resolved through a single free counter, namely the number of coordinates outside the chosen subset. The resulting invariant is the weight enumerator. By Theorem 26 the local weights at genus one span a space of dimension $2^1 = 2$, equal to the size of $\mathcal{L}_1$, so the second stage loses nothing and the genus-one invariant is exactly the first-stage one. By the theorem of Britz [7] the genus-one Tutte polynomial and the family of higher weight enumerators determine one another, so a linear relation among the genus-one Tutte polynomials exists precisely when one exists among the higher weight enumerators.

We verified this directly. For each code and each rank r from 1 to 12 the higher weight enumerator was computed from the dimension table by Möbius inversion: the number of r-dimensional subcodes with support inside A is the Gaussian binomial of d(A) over r, and inversion gives support exactly A. No subcode is enumerated, so the rank-six enumerator, counting more than two hundred billion subcodes, takes seconds. All ranks together give a 9 × 103 integer matrix of rank exactly 8 over $\mathbb{Q}$, whose one-dimensional null space is generated by w without normalisation.

Three further facts about the nine codes were obtained in the course of this verification. None is used in the proof of Theorem 18, and each bears on the open problems of Section 11.

The matrix Θ is a joint distribution. Since ρ(A) = k − d(A^c) by Lemma 4, the exponent pair attached to A equals (d(B), k − |B| + d(B)) with B the complement of A. The column of Θ indexed by (a, b) therefore counts the subsets B of size k − b + a with d(B) = a, so Θ is exactly the joint distribution of the pair (|B|, d(B)) over all subsets. The two descriptions were verified to agree entry by entry. The closed forms of the Remark following Theorem 6 are, in these terms, the counts of the subsets B of sizes 0, 3, 4 and 5.

Matroid decomposability. The circuits of the vector matroid are the minimal supports of nonzero codewords of C⊥ = C, and the genus-one Tutte polynomial is multiplicative over the components. Exactly two of the nine codes decompose: e_8^⊕3 into three components of size eight, and $d_{16} \oplus e_8$ into components of sizes eight and sixteen. The other seven have connected matroids. This rules out an explanation of the relation based solely on multiplicativity over direct-sum components, since seven of the nine terms admit no such factorisation.

Where the relation is forced. Truncating the higher weight enumerators at rank r and recording the rank of the resulting matrix over $\mathbb{Q}$ gives the following.

ranks used	coefficients	spanning dimension	dimension of the relation space
1	6	2	7
1 to 2	16	3	6
1 to 3	33	5	4
1 to 4	49	7	2
1 to 5	62	8	1, generated by w
1 to 12	103	8	1, generated by w

Within this computed sequence of truncations the relation space first becomes one-dimensional at rank five and is unchanged for ranks six to twelve, which contribute forty-one further coefficients and add nothing to the spanning dimension. We use the phrase forced at rank five only as shorthand for this observed stabilisation, not as a structural theorem.

The measured spanning dimensions of the nine codes under successive invariants are as follows.

invariant	spanning dimension	dimension of the relation space
complete weight enumerator, genus 1	2	7
complete weight enumerator, genus 2	3	6
higher weight enumerators, ranks 1 to 12	8	1, generated by w
genus-one Tutte polynomial	8	1, generated by w
complete weight enumerator, genus 6	9	0
genus-two Tutte polynomial	9	0

Genus two. Here $\mathcal{L}_2$ has five members, and by Theorem 26 the local weights span a space of dimension $2^2 = 4$, so exactly one linear relation is present. The second-stage image is therefore a polynomial ring in four independent local weights rather than five. Whether those four separate the nine codes is not settled by the lattice alone; that separation is the content of Theorem 18, which is proved in Section 7 independently of anything in Sections 8 to 10. What the lattice supplies is the combinatorial state space through which the genus-two invariant factors, together with an exact account of the residual algebraic stage.

10. Relation to the invariant theory of self-dual codes

Both ends of the picture are classical and are not claimed here.

Nebe, Rains and Sloane [10] proved that the complete weight enumerators of genus m of Type II binary self-dual codes span the invariant ring of the complex Clifford group of genus m, the case m = 1 being Gleason's theorem [12]. Runge [11] had shown earlier that the homogeneous invariants of degree 2k for that group are spanned by the complete weight enumerators of the binary self-dual codes of length 2k, forming a basis when m ≥ k − 1. Runge's statement concerns the weight-enumerator side and does not by itself determine the separating genus studied here; his sufficient condition is in any case far from sharp for the present question.

length	Runge bound k − 1	least genus for independence
16	7	3
24	11	6
32	15	10

At the Tutte end, Britz [7] proved that the family of higher weight enumerators of a linear code is equivalent to its Tutte polynomial, with an explicit expression of the latter in terms of subcode weights. The verification reported in Section 9, that w generates the relation space of the higher weight enumerators, is a computation within that equivalence and is not an independent theorem.

What is new concerns neither end but the passage between them. The genus-g Tutte polynomial of [1, 2] is a recent object, and we are not aware of a previous description of how much of a complete weight enumerator it resolves. Theorems 17, 20, 22 and 23 give that description: the first stage of the resolution is exactly by the counts of the separation lattice, whose size has the closed form of Theorem 23, and the second is the local-weight evaluation, whose kernel is generated by |ℒ_g| − 2^g independent linear forms.

11. Open problems

The results above suggest the following questions, listed in what appears to be increasing order of difficulty.

1. Determine the least genus for m = 4, that is, for the eighty-five Type II codes of length 32. The algebraic reduction of Section 7.2 applies unchanged, but two distinct routes to the evaluation are available and both are out of reach at that length, for opposite reasons.

The first is the transform of Proposition 15, which needs arrays of 2^n entries: at n = 32 a single 64-bit array requires 32 GiB and several are needed simultaneously, while the three transforms involve $3 \cdot 32 \cdot 2^{31} \approx 2.1 \times 10^{11}$ updates.

The second uses no array at all. On the slice $t_\cup = 1$ with $t_1 = t_2 = t_\cap = 2$, the Lemma of Section 6.3 with m = (1, 1, 1, 0) expresses the value as a complete weight enumerator of genus three, that is, as a sum over triples of codewords with a coordinate-wise factor. Its cost is $|C|^3 = 2^{(3k)}$ in time and negligible in memory. We verified the identity in exact rational arithmetic at n = 8, where both routes give 71037546601802633401.

The two costs at successive lengths are as follows.

n	transform: 3n·2^(n−1) updates	array of 2^n int64	codeword triples: time
24	6.0×10^{8}	128 MiB	$2^{36} \approx 6.9 \times 10^{10}$
32	2.1×10^{11}	32 GiB	$2^{48} \approx 2.8 \times 10^{14}$
40	6.6×10^{13}	8 TiB	$2^{60} \approx 1.2 \times 10^{18}$

At length 24 the transform is the better of the two by two orders of magnitude; at length 32 the first is blocked by memory and the second by time. The obstacle is therefore not a matter of tuning either route, and a third method appears to be required. 2. Explain the relation vector structurally. Theorem 7 gives it in closed form, as the order-six divided difference on the values of A_4 other than 24, so that the vanishing coefficient of $d_{10} \oplus e_7$^⊕2 is the excluded node; but the closed form is a description, not a cause. The natural explanation, that the coefficients of Θ are polynomial in A_4 of degree at most five, is false: only 27 of the 76 columns lie in the corresponding eight-dimensional space. Section 9 narrows the question further: the relation is forced at subcode rank five and not before, and seven of the nine matroids are connected, so no factorisation into components is available for them. What is it about the rank-five subcode statistics that saturates the family at dimension eight, and why is the coefficient of $d_{10} \oplus e_7$^⊕2 zero, that being the unique representative whose weight-four graph has components of odd size, namely 7, 7 and 10? 3. Proposition 11 sharpens the reconstruction bound at genus two from $k \cdot g^2$ to k, by a change of variables that lowers the total degree; at length 24 this replaces 48 by 12, against the lower bound 6 supplied by the value of Nebe. Close the remaining factor of two, and determine whether the analogous reduction holds at genus g, where the modular inequality is available only for pairs.

4. Theorem 27 gives a basis of the linear part of K_g. Describe the whole ideal in each degree: its Hilbert function, and not merely its linear generators, is what governs how much the second stage of the factorisation loses in a given degree, and hence how much of the complete weight enumerator the genus-g polynomial retains at a given length.

5. Determine the rank of the family of lattice-count enumerators of Definition 28 for the Type II codes of a given length, modulo K_g. For g = 1 the first stage alone is the weight enumerator and the answer is Gleason’s theorem. A general answer would convert Corollary 29 from a necessary condition into a predictive statement and would in particular bear on the least genus at length 32.

6. Proposition 19 evaluates the unrestricted eight-variable polynomial in $O(3^n)$ operations rather than the 4^n of Definition 1, a saving by the factor $(4/3)^n$, which is 997 at n = 24; it still falls short of the $O(n \cdot 2^n)$ available on the subvariety. Decide whether $O(n \cdot 2^n)$ is attainable for the unrestricted polynomial, or whether any algorithm of substantially smaller state complexity can avoid the ordered-disjoint-pair representation.

12. Computational details and reproducibility

All computations were carried out in Python 3.12.3 with NumPy 2.4.4, on an x86-64 machine with 8 GiB of available memory, under Linux; the routines are single-threaded and no thread-count environment variables were set. No computer-algebra system is required: the rational rank, the null vector and the minor determinant are recomputed by the supplied exact-integer routines. The proof does not depend on an external database: the nine generator matrices used in every computation are printed in Appendix B. Every numerical value reported in Sections 5 and 7, namely the rank, the relation vector, the minor determinant and the three residues, was recomputed from those printed matrices, parsed back out of the manuscript itself rather than from separately stored representatives, by the routine reproduce of Appendix A; all assertions in that routine passed.

The matrices of Appendix B are those from which the reported values were obtained. Neither the ordering of the nine codes, nor the order of the rows, nor any bit of any matrix was altered after the computation; a coordinate permutation would leave the polynomials unchanged, but the printed data are nevertheless exactly those used.

As a guard against an artefact of the particular modulus, the certificate of Lemma 17 was recomputed at the same point P_1 over the second prime 2147483629, giving the residue 227463606, again nonzero. Both moduli are below 2^31, so the overflow bounds of Section 7.2 apply unchanged.

The computational certificate is cross-checked independently by the closed-form coefficient identities of Section 5, the nonzero minor of Theorem 6, and a recomputation over a second prime. Arithmetic is exact throughout: integer arithmetic for the subset transforms and the dimension lookup, rational and integer arithmetic for the rank computation, the relation of Theorem 6 and the minor determinant, and finite-field arithmetic modulo $2^{31} - 1$ for the certificate of Lemma 17. All arithmetic used in the mathematical quantities reported in this paper is integer, rational or finite-field; no floating-point arithmetic enters any of them.

Approximate resource use, on the reference machine above. These timings are reproducibility information and are not used in any mathematical claim.

computation	time	peak memory
one dimension table d(·) at n = 24	≈ 5 s	≈ 130 MB
all nine genus-one coefficient lists	≈ 60 s	≈ 260 MB
exact rank and relation (9 × 76 over $\mathbb{Q}$)	< 1 s	negligible
one genus-two evaluation, nine codes	≈ 87 s	≈ 400 MB

References

[1] T. Miezaki, M. Oura, T. Sakuma and H. Shinohara, *The Tutte polynomials of genus g*, arXiv:2402.06845.

[2] T. Miezaki, M. Oura, T. Sakuma and H. Shinohara, *A generalization of the Tutte polynomials*, Proc. Japan Acad. Ser. A Math. Sci. **95** (2019), 111–113.

[3] G. Nebe, *Kneser–Hecke operators in coding theory*, Abh. Math. Semin. Univ. Hambg. **76** (2006), 79–90.

[4] C. Greene, *Weight enumeration and the geometry of linear codes*, Stud. Appl. Math. **55** (1976), 119–128.

[5] V. Pless and N. J. A. Sloane, *On the classification and enumeration of self-dual codes*, J. Combin. Theory Ser. A **18** (1975), 313–335.

[6] T. Britz and K. Shiromoto, *A MacWilliams type identity for matroids*, Discrete Math. **308** (2008), no. 20, 4551–4559.

[7] T. Britz, *Code enumerators and Tutte polynomials*, IEEE Trans. Inform. Theory **56** (2010), no. 9, 4350–4358.

[8] T. Britz, K. Shiromoto and T. Westerbäck, *Demi-matroids from codes over finite Frobenius rings*, Des. Codes Cryptogr. **75** (2015), no. 1, 97–107.

[9] H. S. Chakraborty, T. Miezaki and M. Oura, *Harmonic Tutte polynomials of matroids*, Des. Codes Cryptogr. **91** (2023), 2223–2236.

[10] G. Nebe, E. M. Rains and N. J. A. Sloane, *Self-Dual Codes and Invariant Theory*, Springer, 2006.

[11] B. Runge, *Codes and Siegel modular forms*, Discrete Math. **148** (1996), 175–204.

[12] A. M. Gleason, *Weight polynomials of self-dual codes and the MacWilliams identities*, Actes du Congrès International des Mathématiciens 1970, Tome 3, 211–215.

[13] N. J. A. Sloane, editor, *The On-Line Encyclopedia of Integer Sequences*, sequence A006896, published electronically at oeis.org, accessed 2026.

Appendix A. Reference implementation

Exact integer, rational and finite-field arithmetic only. Run as a script, this file re-derives from the matrices of Appendix B, and asserts, every numerical value stated in Sections 5, 7, 8 and 9. It also contains routines not used in any proof: moment_vanishing and divided_difference_form for Theorem 7; reduced_exponents for Proposition 11 and interpolation_matrix_rank for its nonsingularity certificate; genus2_slice_by_triples and genus2_full_by_disjoint_pairs, which implement the alternative algorithms of Open Problem 1 and Proposition 19 and do not reach length 32 within the reported resources; matroid_components and cumulative_hwe_rank for Section 9; and rank3_evaluate with rank3_all, which evaluate the rank-three column-multiset distribution referred to in Open Problem 2, in about one hour for the nine codes.

```
"""Reference implementation.  Integer, rational and finite-field arithmetic o
nly.
```

```
Parameterised by (n, k), but the dense 2^n arrays and machine-word masks are
intended only for feasible sizes; every reported computation is at n = 24, k
= 12.
Run as a script it re-derives and asserts every value stated in the paper."""

import numpy as np
from fractions import Fraction

PRIME = (1 << 31) - 1


# ---------------------------------------------------------------- input

def parse_rows(bit_strings, n, k):
    """Convert the printed Appendix B rows to integers, with validation."""
    assert len(bit_strings) == k, "expected k generator rows"
    assert all(len(s) == n for s in bit_strings), "each row must have n bits"
    assert all(set(s) <= {"0", "1"} for s in bit_strings), "non-binary digit"
    return [int(s, 2) for s in bit_strings]


def codewords(G):
    c = [0]
    for r in G:
        c = c + [x ^ r for x in c]
    return c


def checks(G, n, k):
    # With n = 2k, full rank together with self-orthogonality implies self-du
ality.
    assert n == 2 * k, "self-duality requires n = 2k"
    assert len(G) == k, "wrong number of generator rows"
    assert all(0 <= r < (1 << n) for r in G), "generator row outside F_2^n"
    assert len(set(codewords(G))) == 1 << k, "not full rank"
    assert all(bin(a & b).count("1") % 2 == 0 for a in G for b in G), "not se
lf-orthogonal"
    assert all(bin(c).count("1") % 4 == 0 for c in codewords(G)), "not doubly
even"
    return True


# ---------------------------------------------------------------- transforms

def dim_table(G, n, k):
    """d(A) for every A, by a subset-sum transform; exact integers only."""
    N = 1 << n
    cnt = np.zeros(N, dtype=np.int64)
```

```
    for c in codewords(G):
        m = 0
        for j in range(n):
            if (c >> (n - 1 - j)) & 1:
                m |= 1 << j
        cnt[m] += 1
    for j in range(n):
        v = cnt.reshape(N >> (j + 1), 2, 1 << j)
        v[:, 1, :] += v[:, 0, :]
    lut = np.zeros((1 << k) + 1, dtype=np.int64)      # N(A) = 2^{d(A)}
    for j in range(k + 1):
        lut[1 << j] = j
    return lut[cnt]


def sizes(n):
    N = 1 << n
    idx = np.arange(N, dtype=np.uint32)
    s = np.zeros(N, dtype=np.int64)
    for j in range(n):
        s += (idx >> j) & 1
    return s


# ------------------------------------------------------------------ genus one

def genus1(G, n, k):
    """Coefficient dictionary of R^(1), keyed by (rho(E)-rho(A))*(k+1) + d(A)
."""
    d = dim_table(G, n, k)
    s = sizes(n)
    rho = s - d
    rE = int(rho[-1])
    key = (rE - rho) * (k + 1) + (s - rho)
    u, c = np.unique(key, return_counts=True)
    return dict(zip(u.tolist(), c.tolist())), rE


def rank_and_relation(dists):
    """Exact rational Gauss-Jordan; returns rank, column count, relation basi
s."""
    keys = sorted(set().union(*[set(x) for x in dists]))
    m = len(dists)
    aug = [[Fraction(x.get(kk, 0)) for kk in keys]
           + [Fraction(int(i == j)) for j in range(m)]
           for i, x in enumerate(dists)]
    r = 0
    for c in range(len(keys)):
        piv = next((i for i in range(r, m) if aug[i][c] != 0), None)
```

```
        if piv is None:
            continue
        aug[r], aug[piv] = aug[piv], aug[r]
        p = aug[r][c]
        aug[r] = [x / p for x in aug[r]]
        for i in range(m):
            if i != r and aug[i][c] != 0:
                f = aug[i][c]
                aug[i] = [a - f * b for a, b in zip(aug[i], aug[r])]
        r += 1
        if r == m:
            break
    rel = [row[len(keys):] for row in aug if all(v == 0 for v in row[:len(key
s)])]
    return r, keys, rel


def det_bareiss(a):
    """Fraction-free integer determinant, independent of the rational routine
."""
    m = [row[:] for row in a]
    N = len(m)
    prev, sign = 1, 1
    for c in range(N - 1):
        if m[c][c] == 0:
            p = next((i for i in range(c + 1, N) if m[i][c] != 0), None)
            if p is None:
                return 0
            m[c], m[p] = m[p], m[c]
            sign = -sign
        for i in range(c + 1, N):
            for j in range(c + 1, N):
                m[i][j] = (m[i][j] * m[c][c] - m[i][c] * m[c][j]) // prev
        prev = m[c][c]
    return sign * m[N - 1][N - 1]


# ---------------------------------------------------------------- genus two

def _powers(x, m):
    t = np.ones(m + 1, dtype=np.int64)
    for i in range(1, m + 1):
        t[i] = t[i - 1] * x % PRIME
    return t


def genus2_slice(d, n, k, xi, eta, zeta, t1, t2, tc):
    """Normalised R^(2) on the slice t_cup = 1, modulo PRIME, in O(n 2^n).
    Chunked accumulation keeps every unreduced int64 sum below 2^63."""
```

```
    N = 1 << n
    s = sizes(n)

    def zsup(a):
        for j in range(n):
            v = a.reshape(N >> (j + 1), 2, 1 << j)
            v[:, 0, :] += v[:, 1, :]
            a %= PRIME
        return a

    def mob(a):
        for j in range(n):
            v = a.reshape(N >> (j + 1), 2, 1 << j)
            v[:, 1, :] -= v[:, 0, :]
            a %= PRIME
        return a

    F1 = zsup((_powers(xi, n)[s] * _powers(t1, k)[d]) % PRIME)
    F2 = zsup((_powers(eta, n)[s] * _powers(t2, k)[d]) % PRIME)
    H = mob((_powers(zeta, n)[s] * _powers(tc, k)[d]) % PRIME)
    acc, CH = 0, 1 << 21
    for i in range(0, N, CH):
        blk = H[i:i + CH] * F1[i:i + CH] % PRIME * F2[i:i + CH] % PRIME
        acc = (acc + int(blk.sum() % PRIME)) % PRIME
    return acc




# ============================================================================
=
#  SEPARATION LATTICE  (Sections 8 and 9)
#  The local weight W_g(s) depends on the pattern s only through the admissib
le
#  set A(s), the intersection of the state sets attached to the constraints t
hat
#  s activates.  The distinct admissible sets form the separation lattice L_g
.
# ============================================================================
=

def state_sets(g):
    """The g^2 state sets A_lambda inside F_2^g: g singletons, C(g,2)
    intersections, C(g,2) unions."""
    import itertools
    S = list(itertools.product((0, 1), repeat=g))
    out = []
    for i in range(g):
        out.append(frozenset(a for a in S if a[i] == 1))
```

```
    for i, j in itertools.combinations(range(g), 2):
        out.append(frozenset(a for a in S if a[i] == 1 and a[j] == 1))
        out.append(frozenset(a for a in S if a[i] == 1 or a[j] == 1))
    assert len(out) == g * g
    return S, out


def separation_lattice(g):
    """Closure under intersection of the state sets, starting from all of F_2
^g."""
    S, sets = state_sets(g)
    lat = {frozenset(S)}
    changed = True
    while changed:
        changed = False
        new = set()
        for X in lat:
            for A in sets:
                Y = frozenset(X & A)
                if Y not in lat:
                    new.add(Y)
        if new:
            lat |= new
            changed = True
    return lat


def lattice_size_formula(g):
    """sum_r C(g,r) 2^C(r,2): labelled graphs on at most g vertices."""
    from math import comb
    return sum(comb(g, r) * 2 ** (r * (r - 1) // 2) for r in range(g + 1))


def admissible_set(s, sets, S):
    """A(s) = intersection of the state sets activated by the pattern s."""
    cur = frozenset(S)
    for bit, A in zip(s, sets):
        if bit:
            cur = cur & A
    return cur


def level_set_test(g, seed=7):
    """With generic state weights, W_g(s) = W_g(s') if and only if A(s) = A(s
').
    Returns (forward, converse, number of classes)."""
    import itertools, random
    random.seed(seed)
    S, sets = state_sets(g)
```

```
    wt = {a: Fraction(random.randint(1, 10 ** 6)) for a in S}
    byA, byW = {}, {}
    for s in itertools.product((0, 1), repeat=g * g):
        A = admissible_set(s, sets, S)
        W = sum(wt[a] for a in A)
        byA.setdefault(A, set()).add(W)
        byW.setdefault(W, set()).add(A)
    return (all(len(v) == 1 for v in byA.values()),
            all(len(v) == 1 for v in byW.values()),
            len(byA))




def local_weight_span(g):
    """The local weights span the full 2^g-dimensional space of linear forms
in
    the state weights.  Returns (|L_g|, rank, number of relations)."""
    import itertools
    S = list(itertools.product((0, 1), repeat=g))
    mem = sorted(separation_lattice(g), key=lambda x: (-len(x), sorted(x)))
    M = [[Fraction(1) if a in m else Fraction(0) for a in S] for m in mem]
    r, n, rk = 0, len(S), 0
    rows = [row[:] for row in M]
    for c in range(n):
        p = next((i for i in range(rk, len(rows)) if rows[i][c] != 0), None)
        if p is None:
            continue
        rows[rk], rows[p] = rows[p], rows[rk]
        pv = rows[rk][c]
        rows[rk] = [x / pv for x in rows[rk]]
        for i in range(len(rows)):
            if i != rk and rows[i][c] != 0:
                f = rows[i][c]
                rows[i] = [a - f * b for a, b in zip(rows[i], rows[rk])]
        rk += 1
        if rk == len(rows):
            break
    return len(mem), rk, len(mem) - rk


# ============================================================================
=
#  HIGHER WEIGHT ENUMERATORS IN O(n 2^n)  (Section 9)
#  The number of r-dimensional subcodes with support contained in A is the
#  Gaussian binomial [d(A) choose r]_2; Mobius inversion over the subset latt
ice
#  gives support exactly A.  No subcode is enumerated.
# ============================================================================
```

```
=

def gaussian_binomial(m, r):
    if r > m:
        return 0
    num = den = 1
    for i in range(r):
        num *= (1 << m) - (1 << i)
        den *= (1 << r) - (1 << i)
    return num // den


def higher_weight_enumerator(d, r, n=24, k=12):
    """Support enumerator of the r-dimensional subcodes, keyed by support siz
e."""
    tab = np.array([gaussian_binomial(m, r) for m in range(k + 1)], dtype=np.
int64)
    f = tab[d].copy()
    for j in range(n):
        v = f.reshape((1 << n) >> (j + 1), 2, 1 << j)
        v[:, 1, :] -= v[:, 0, :]
    sz = sizes(n)
    out = {}
    for u in range(n + 1):
        t = int(f[sz == u].sum())
        if t:
            out[u] = t
    return out




# ============================================================================
=
#  A MEMORY-FREE ROUTE, AND THE RANK-THREE OPEN CASE  (Open problems 1 and 5)
# ============================================================================
=




# ============================================================================
=
#  THE RELATION VECTOR IN CLOSED FORM, AND THE SHARPENED RECONSTRUCTION BOUND
# ============================================================================
=

A4_VALUES = [0, 6, 12, 18, 24, 30, 42, 42, 66]
```

```
def moment_vanishing(w=None, A4=None):
    """Orders r for which sum_i w_i A4_i^r vanishes, and the first that does
not."""
    w = W_PUBLISHED if w is None else w
    A4 = A4_VALUES if A4 is None else A4
    out = []
    r = 0
    while True:
        s = sum(w[i] * A4[i] ** r for i in range(len(w)))
        out.append((r, s))
        if s != 0 or r > 12:
            return out
        r += 1


def divided_difference_form():
    """After combining the two codes with equal A4, the relation is the order
-six
    divided difference on the seven remaining values of A4."""
    vals = [0, 6, 12, 18, 24, 30, 42, 66]
    u = [576, -2772, 4928, -3465, 0, 924, 99 + (-297), 7]
    keep = [v for v in vals if v != 24]
    c = {}
    for v in keep:
        p = Fraction(1)
        for x in keep:
            if x != v:
                p *= (v - x)
        c[v] = Fraction(1) / p
    lead = c[keep[0]] / u[0]
    ok = all(c[v] / lead == u[vals.index(v)] for v in keep)
    return ok, u


def reduced_exponents(d1, d2, dc, du):
    """The change of variables of Proposition 11; all four are non-negative a
nd
    their sum is d(U), hence at most k."""
    return (dc, d1 - dc, d2 - dc, du - d1 - d2 + dc)



def interpolation_matrix_rank(k, p=(1 << 61) - 1):
    """Nonsingularity of the interpolation matrix of Proposition 11.
    V[m,e] = 2^<n(m), e> with n(m) = (|m|, m1+mu, m2+mu, mu), m and e ranging
    over { x >= 0 : |x| <= k }.  A full rank modulo p certifies nonsingularit
y
    over Q.  Returns (size, rank)."""
```

```
    import itertools
    S = [x for x in itertools.product(range(k + 1), repeat=4) if sum(x) <= k]
    N = len(S)
    M = [[pow(2, sum(x) * e[0] + (m[0] + m[3]) * e[1] + (m[1] + m[3]) * e[2]
+ m[3] * e[3], p)
          for e in S] for m, x in ((m, m) for m in S)]
    r = 0
    for c in range(N):
        piv = next((i for i in range(r, N) if M[i][c] % p), None)
        if piv is None:
            continue
        M[r], M[piv] = M[piv], M[r]
        inv = pow(M[r][c], p - 2, p)
        M[r] = [v * inv % p for v in M[r]]
        for i in range(r + 1, N):
            f = M[i][c] % p
            if f:
                M[i] = [(a - f * b) % p for a, b in zip(M[i], M[r])]
        r += 1
    return N, r


def matroid_components(G, n=24):
    """Connected components of the vector matroid; the circuits are the minim
al
    supports of nonzero codewords of C^perp = C.  T^(1) is multiplicative her
e."""
    code = [c for c in codewords(G) if c]
    circ = [c for c in code if not any(o != c and (o | c) == c for o in code)
]
    p = list(range(n))

    def find(x):
        while p[x] != x:
            p[x] = p[p[x]]
            x = p[x]
        return x

    for c in circ:
        s = [j for j in range(n) if (c >> (n - 1 - j)) & 1]
        for j in s[1:]:
            p[find(j)] = find(s[0])
    from collections import Counter
    return sorted(Counter(find(j) for j in range(n)).values()), len(circ)


def cumulative_hwe_rank(tables, upto, n=24, k=12):
    """Rank of the nine codes under the higher weight enumerators of ranks
    1..upto, and the relation space if it is one-dimensional."""
```

```
    cols = []
    for r in range(1, upto + 1):
        Ar = [higher_weight_enumerator(tables[i], r, n, k) for i in range(len
(tables))]
        for kk in sorted(set().union(*[set(x) for x in Ar])):
            cols.append([Ar[i].get(kk, 0) for i in range(len(tables))])
    M = [{j: cols[j][i] for j in range(len(cols))} for i in range(len(tables)
)]
    rk, _, rel = rank_and_relation(M)
    return len(cols), rk, rel


def lattice_member(T, edges, g):
    """The member A(T, G) of the separation lattice: the states that are one
on
    T and satisfy a_i or a_j for every edge of the graph G on the free indice
s."""
    import itertools
    S = list(itertools.product((0, 1), repeat=g))
    return frozenset(a for a in S
                     if all(a[i] == 1 for i in T)
                     and all(a[i] == 1 or a[j] == 1 for i, j in edges))


def relation_expansion(T, edges):
    """Coefficients of the indicator of A(T, G) in the basis {1_{A_U}}, by th
e
    edgewise expansion of Theorem 27."""
    coef = {frozenset(T): Fraction(1)}
    for (i, j) in edges:
        new = {}
        for U, c in coef.items():
            for add, sgn in ((frozenset({i}), 1), (frozenset({j}), 1),
                             (frozenset({i, j}), -1)):
                V = U | add
                new[V] = new.get(V, Fraction(0)) + c * sgn
        coef = {u: v for u, v in new.items() if v != 0}
    return coef


def check_relation_basis(g):
    """Verify the expansion of Theorem 25 on every state, for every member."""
"
    import itertools
    S = list(itertools.product((0, 1), repeat=g))
    seen = set()
    count = 0
    for r in range(g + 1):
        for T in itertools.combinations(range(g), r):
```

```
            free = [i for i in range(g) if i not in T]
            pairs = list(itertools.combinations(free, 2))
            for mask in range(1 << len(pairs)):
                edges = [pairs[b] for b in range(len(pairs)) if (mask >> b) &
1]
                A = lattice_member(T, edges, g)
                seen.add(A)
                coef = relation_expansion(T, edges)
                for a in S:
                    lhs = 1 if a in A else 0
                    rhs = sum(c for U, c in coef.items() if all(a[i] == 1 for
i in U))
                    assert lhs == rhs, (T, edges, a)
                if edges:
                    count += 1
    assert len(seen) == lattice_size_formula(g)
    assert count == lattice_size_formula(g) - 2 ** g
    return len(seen), count


def genus2_slice_by_triples(G, xi, eta, zeta, n=24, k=12):
    """The same value as genus2_slice at t = 2, computed as a sum over triple
s of
    codewords: no array of size 2^n is used.  Cost |C|^3.  Exhibits the
    time-memory trade-off of Open Problem 1; validation only."""
    code = codewords(G)
    MASK = (1 << n) - 1
    tab = {}
    for a in (0, 1):
        for b in (0, 1):
            for p in (0, 1):
                tot = 0
                for s1, s2 in ((0, 0), (1, 0), (0, 1), (1, 1)):
                    if (a or p) and not s1:
                        continue
                    if (b or p) and not s2:
                        continue
                    tot += {(0, 0): 1, (1, 0): xi, (0, 1): eta,
                            (1, 1): xi * eta * zeta}[(s1, s2)]
                tab[(a, b, p)] = tot
    total = 0
    for c1 in code:
        for c2 in code:
            for c0 in code:
                v = 1
                for i in range(n):
                    v *= tab[((c1 >> (n - 1 - i)) & 1,
                              (c2 >> (n - 1 - i)) & 1,
                              (c0 >> (n - 1 - i)) & 1)]
                total += v
```

```
    return total



def genus2_full_by_disjoint_pairs(G, xi, eta, ze, ta, t1, t2, tc, tu, n=24, k
=12):
    """The full eight-variable genus-two function by a double Mobius transfor
m.
    Since A1 u A2 is the complement of A1^c n A2^c, transforming both couplin
g
    factors leaves a sum over ordered pairs of DISJOINT subsets: 3^n, not 4^n
.
    See Proposition 19.  Written for clarity, not speed."""
    N = 1 << n
    MASK = N - 1
    d = dim_table(G, n, k)
    sz = sizes(n)
    f1 = [xi ** int(sz[A]) * t1 ** int(d[A]) for A in range(N)]
    f2 = [eta ** int(sz[A]) * t2 ** int(d[A]) for A in range(N)]
    h = [ze ** int(sz[A]) * tc ** int(d[A]) for A in range(N)]
    kk = [ta ** (n - int(sz[J])) * tu ** int(d[MASK ^ J]) for J in range(N)]

    def mob(v):
        v = list(v)
        for j in range(n):
            for m in range(N):
                if (m >> j) & 1:
                    v[m] = v[m] - v[m ^ (1 << j)]
        return v

    hh, kh = mob(h), mob(kk)
    total = 0
    for S in range(N):
        if hh[S] == 0:
            continue
        rest = MASK ^ S
        sub = rest
        while True:
            Sp = sub
            if kh[Sp] != 0:
                F1 = sum(f1[A] for A in range(N) if (A & S) == S and (A & Sp)
== 0)
                F2 = sum(f2[A] for A in range(N) if (A & S) == S and (A & Sp)
== 0)
                total += hh[S] * kh[Sp] * F1 * F2
            if sub == 0:
                break
            sub = (sub - 1) & rest
```

```
    return total


def rank3_evaluate(G, seed=1, batch=256, n=24, progress=False):
    """Random multiplicative evaluation of the rank-three column-multiset
    distribution, modulo PRIME.  Enumerating the planes <a,b>, the four cells
of
    a plane give mu by four popcounts against the third generator; a test
    function phi(mu) = prod_j r_j^{mu_j} then factorises.  Run for all nine c
odes
    and form sum_i w_i * value_i: zero means the relation survives at rank th
ree.
    About seven minutes per code."""
    import time
    MASK = (1 << n) - 1
    POPT = np.zeros(1 << 12, dtype=np.int8)
    for j in range(12):
        POPT[(np.arange(1 << 12) >> j) & 1 == 1] += 1

    def pc(a):
        a = a.astype(np.int64)
        return POPT[a & 0xFFF] + POPT[(a >> 12) & 0xFFF]

    rng = np.random.default_rng(seed)
    r = rng.integers(2, PRIME - 2, 8)
    code = np.array(sorted(codewords(G)), dtype=np.int64)
    nz = code[code != 0]
    a_all = nz[:, None]
    b_all = nz[None, :]
    keep = (b_all > a_all) & (np.bitwise_xor(a_all, b_all) > b_all)
    ia, ib = np.nonzero(keep)
    A, B = nz[ia], nz[ib]
    inv = [pow(int(r[j]), PRIME - 2, PRIME) for j in range(4)]
    q = [int(r[j + 4]) * inv[j] % PRIME for j in range(4)]
    QT = [_powers(q[j], n) for j in range(4)]
    RT = [_powers(int(r[j]), n) for j in range(4)]
    total, t0 = 0, time.time()
    for s in range(0, len(A), batch):
        a = A[s:s + batch][:, None]
        b = B[s:s + batch][:, None]
        cells = np.concatenate([(~a) & (~b) & MASK, (~a) & b & MASK,
                                a & (~b) & MASK, a & b], axis=1)
        sz = pc(cells).astype(np.int64)
        pre = np.ones(len(sz), dtype=np.int64)
        for j in range(4):
            pre = pre * RT[j][sz[:, j]] % PRIME
        acc = np.ones((len(sz), code.size), dtype=np.int64)
        for j in range(4):
            it = pc(cells[:, j][:, None] & code[None, :]).astype(np.int64)
```

```
            acc = acc * QT[j][it] % PRIME
        total = (total + int((acc.sum(axis=1) % PRIME * pre % PRIME).sum() %
PRIME)) % PRIME
        if progress and (s // batch) % 400 == 0:
            print(f"      planes {s:,}/{len(A):,}  [{time.time()-t0:.0f}s]",
flush=True)
    return total * pow(4, PRIME - 2, PRIME) % PRIME


def rank3_all(appendix_B, seed=1, n=24, k=12):
    """Settles open problem 5.  About one hour for the nine codes."""
    G = [parse_rows(rows, n, k) for rows in appendix_B]
    vals = []
    for i, g in enumerate(G):
        v = rank3_evaluate(g, seed=seed, n=n)
        vals.append(v)
        print(f"  code {i + 1}: {v}", flush=True)
    res = sum(a * b for a, b in zip(W_PUBLISHED, vals)) % PRIME
    print(f"  sum w_i * value_i = {res} mod {PRIME}")
    print("  " + ("VANISHES: the relation survives at rank three"
                  if res == 0 else
                  "NONZERO: the relation does not survive at rank three"))
    return res


# ------------------------------------------------ reproduction of the paper

W_PUBLISHED = [576, -2772, 4928, -3465, 0, 924, 99, -297, 7]
MINOR_COLUMNS = [(0, 8), (3, 6), (4, 6), (5, 2), (5, 4), (5, 5), (8, 2), (12,
0)]
MINOR_DET = 14739701760
P1 = (2, 3, 5, 7, 11, 13)
RESIDUE_P1 = 749926473


def reproduce(appendix_B, n=24, k=12, do_genus2=True):
    """appendix_B: list of nine lists of k bit-strings, in the order of Table
1."""
    G = [parse_rows(rows, n, k) for rows in appendix_B]
    assert len(G) == 9
    for g in G:
        assert checks(g, n, k)

    dists = [genus1(g, n, k)[0] for g in G]
    r, keys, rel = rank_and_relation(dists)
    assert len(keys) == 76, "expected 76 monomials"
    assert r == 8, "expected rank 8"

    from math import gcd
```

```
    v = rel[0]
    L = 1
    for x in v:
        L = L * x.denominator // gcd(L, x.denominator)
    w = [int(x * L) for x in v]
    g0 = 0
    for x in w:
        g0 = gcd(g0, abs(x))
    w = [x // g0 for x in w]
    if w[0] < 0:
        w = [-x for x in w]
    assert w == W_PUBLISHED, "relation vector differs from the published one"
    assert all(sum(w[i] * dists[i].get(kk, 0) for i in range(9)) == 0 for kk
in keys)

    # closed-form entries of Theta, provable by hand (see the Remark after Th
eorem 6)
    from math import comb
    for i, g in enumerate(G):
        A4 = sum(1 for c in codewords(g) if bin(c).count("1") == 4)
        row = dists[i]
        assert row.get(12 * (k + 1) + 0, 0) == 1
        assert sum(row.values()) == 1 << n
        assert row.get(0 * (k + 1) + 9, 0) == comb(n, 21)
        assert row.get(0 * (k + 1) + 8, 0) == comb(n, 20) - A4
        assert row.get(1 * (k + 1) + 8, 0) == 20 * A4
        assert row.get(0 * (k + 1) + 7, 0) == comb(n, 19) - 20 * A4

    cols = [keys.index(a * (k + 1) + b) for (a, b) in MINOR_COLUMNS]
    sub = [[dists[i].get(keys[j], 0) for j in cols] for i in range(8)]
    assert det_bareiss(sub) == MINOR_DET, "minor determinant differs"

    # separation lattice: closure equals the closed formula, and the level-se
t
    # property holds, for g = 1..4
    for gg in range(1, 5):
        assert len(separation_lattice(gg)) == lattice_size_formula(gg)
        fwd, conv, cls = level_set_test(gg)
        assert fwd and conv and cls == lattice_size_formula(gg)
    assert [lattice_size_formula(gg) for gg in range(1, 7)] == [2, 5, 18, 113
, 1450, 40069]

    # the local weights span exactly 2^g dimensions, so the second-stage kern
el
    # is generated by |L_g| - 2^g independent linear forms
    for gg in range(1, 5):
        assert check_relation_basis(gg) == (lattice_size_formula(gg),
                                            lattice_size_formula(gg) - 2 ** g
g)
```

```
        size, rk, rels = local_weight_span(gg)
        assert rk == 2 ** gg, (gg, rk)
        assert rels == lattice_size_formula(gg) - 2 ** gg

    # higher weight enumerators: w is the unique relation, ranks 1..12
    tables = [dim_table(g, n, k) for g in G]
    cols = []
    for rr in range(1, k + 1):
        Ar = [higher_weight_enumerator(tables[i], rr, n, k) for i in range(9)
]
        for kk in sorted(set().union(*[set(x) for x in Ar])):
            cols.append([Ar[i].get(kk, 0) for i in range(9)])
    assert len(cols) == 103, len(cols)
    M = [{j: cols[j][i] for j in range(len(cols))} for i in range(9)]
    rh, _, relh = rank_and_relation(M)
    assert rh == 8, rh
    vh = relh[0]
    L = 1
    for x in vh:
        L = L * x.denominator // gcd(L, x.denominator)
    wh = [int(x * L) for x in vh]
    g1 = 0
    for x in wh:
        g1 = gcd(g1, abs(x))
    wh = [x // g1 for x in wh]
    if wh[0] < 0:
        wh = [-x for x in wh]
    assert wh == W_PUBLISHED, "the higher-weight-enumerator relation differs
from w"

    if do_genus2:
        vals = [genus2_slice(d, n, k, *P1) for d in tables]
        res = sum(a * b for a, b in zip(w, vals)) % PRIME
        assert res == RESIDUE_P1, "residue differs from the published certifi
cate"
        assert res != 0
    return True
```

Appendix B. The nine generator matrices

Each matrix is given as twelve rows of twenty-four bits, in the ordering of Table 1. The routine parse_rows of Appendix A converts these rows to integers and validates their shape.

g24

```
100000000000111110010100
010000000000001110111001
001000000000011101110010
000100000000101101101100
```

```
000010000000011100001111
000001000000101111000011
000000100000100100110111
000000010000110110101010
000000001000111011111111
000000000100110101011001
000000000010000111011110
000000000001010111100101
```

d4^⊕6

```
100000000000101111111111
010000000000011110000111
001000000000111010011010
000100000000111000110101
000010000000010100111011
000001000000010110111001
000000100000110010011110
000000010000011101011100
000000001000111000101101
000000000100011110100110
000000000010110011100011
000000000001110100000000
```

d6^⊕4

```
100000000000001111001011
010000000000110110100011
001000000000000010101000
000100000000010110100111
000010000000101011011100
000001000000100101010111
000000100000101000101111
000000010000001010100000
000000001000110000000100
000000000100101100101101
000000000010101100101110
000000000001001110011011
```

d8^⊕3

```
100000000000100000001010
010000000000011001001111
001000000000010111100101
000100000000000000011010
000010000000110011010011
000001000000001011000000
000000100000001110101110
000000010000011101001011
000000001000011010000000
000000000100001010000001
```

000000000010011001101011
000000000011110011011001

d10⊕e7^⊕2

100000000000111011001100
010000000000110011011100
001000000000001010010000
000100000000101101010101
000010000000011000010000
000001000000101101110100
000000100000000001000110
000000010000100000000110
000000001000100001000010
000000000100000100100001
000000000010001000011000
000000000001101001110101

d12^⊕2

100000000000000010100001
010000000000100001010000
001000000000111010111000
000100000000100101100111
000010000000000010100010
000001000000100001001000
000000100000100111000111
000000010000000010100100
000000001000110001000000
000000000100011011111000
000000000010101001000000
000000000001000110100000

e8^⊕3

111111110000000000000000
000011110000000000000000
001100110000000000000000
010101010000000000000000
000000001111111100000000
000000000000111100000000
000000000011001100000000
000000000101010100000000
000000000000000011111111
000000000000000000001111
000000000000000000110011
000000000000000001010101

d16⊕e8

100000000000111001111000
010000000000000110000100

```
001000000000000001100001
000100000000100000100001
000010000000000000110001
000001000000000010000110
000000100000000000101001
000000010000001000100001
000000001000000100000110
000000000100111001011001
000000000010000110000010
000000000001010000100001
```

d24+

```
111100000000000000000000
110011000000000000000000
110000110000000000000000
110000001100000000000000
110000000011000000000000
110000000000110000000000
110000000000001100000000
110000000000000011000000
110000000000000000110000
110000000000000000001100
110000000000000000000011
101010101010101010101010
```

Appendix C. The coefficient matrix Θ

The 9 × 76 integer matrix Θ of Section 5, with rows in the ordering of Table 1 and columns in the lexicographic ordering of the exponent pairs (ρ(E) − ρ(A), |A| − ρ(A)). With these 684 integers the reader may verify $\Theta^{T}w = 0$ in all 76 coordinates and evaluate the 8 × 8 minor of Theorem 6 without executing any program.

The 76 exponent pairs, in order:

```
(0,0) (0,1) (0,2) (0,3) (0,4) (0,5) (0,6) (0,7) (0,8) (0,9)
(0,10) (0,11) (0,12) (1,0) (1,1) (1,2) (1,3) (1,4) (1,5) (1,6)
(1,7) (1,8) (1,9) (2,0) (2,1) (2,2) (2,3) (2,4) (2,5) (2,6)
(2,7) (2,8) (3,0) (3,1) (3,2) (3,3) (3,4) (3,5) (3,6) (3,7)
(3,8) (4,0) (4,1) (4,2) (4,3) (4,4) (4,5) (4,6) (4,7) (4,8)
(5,0) (5,1) (5,2) (5,3) (5,4) (5,5) (6,0) (6,1) (6,2) (6,3)
(6,4) (7,0) (7,1) (7,2) (7,3) (7,4) (8,0) (8,1) (8,2) (8,3)
(8,4) (9,0) (9,1) (10,0) (11,0) (12,0)
```

Row 1, g24

```
   1391040    2071104    1870176    1295360     734712     346104
    134596      42504      10626       2024        276         24
         1    2071104    1277696     425040      91080      12144
       759          0          0          0          0    1870176
    425040      35420          0          0          0          0
```

0	0	1295360	91080	0	0
0	0	0	0	0	734712
12144	0	0	0	0	0
0	0	346104	759	0	0
0	0	134596	0	0	0
0	42504	0	0	0	0
10626	0	0	0	0	2024
0	276	24	1		

Row 2, d4^⊕6

982784	1676544	1646976	1203200	705696	339264
133456	42384	10620	2024	276	24
1	1676544	1536192	791040	311040	104064
29760	6840	1140	120	6	1646976
791040	184080	28560	3240	240	15
0	0	1203200	311040	28560	1100
0	0	0	0	0	705696
104064	3240	0	0	0	0
0	0	339264	29760	240	0
0	0	133456	6840	15	0
0	42384	1140	0	0	0
10620	120	0	0	0	2024
6	276	24	1		

Row 3, d6^⊕4

714752	1367040	1453440	1118336	677952	332568
132324	42264	10614	2024	276	24
1	1367040	1601792	1031040	486576	185040
56853	13464	2268	240	12	1453440
1031040	376704	97272	21204	4128	666
72	4	1118336	486576	97272	10902
792	36	0	0	0	677952
185040	21204	792	6	0	0
0	0	332568	56853	4128	36
0	0	132324	13464	666	0
0	42264	2268	72	0	0
10614	240	4	0	0	2024
12	276	24	1		

Row 4, d8^⊕3

539136	1125888	1286400	1040384	651456	326016
131200	42144	10608	2024	276	24
1	1125888	1556864	1174272	623232	255744
82080	19872	3384	360	18	1286400
1174272	573728	191520	51120	11328	1932
216	12	1040384	623232	191520	34248
4464	504	48	3	0	651456
255744	51120	4464	180	0	0

0	0	326016	82080	11328	504
0	0	131200	19872	1932	48
0	42144	3384	216	3	0
10608	360	12	0	0	2024
18	276	24	1		

Row 5, d10 ⊕ e7^⊕2

401408	928256	1138368	967680	625968	319592
130084	42024	10602	2024	276	24
1	928256	1492736	1263808	734104	319088
105861	26092	4488	480	24	1138368
1263808	734608	290248	86450	20384	3603
418	24	967680	734104	290248	74370
13784	2333	352	39	2	625968
319088	86450	13784	1034	48	1
0	0	319592	105861	20384	2333
48	0	130084	26092	3603	352
1	42024	4488	418	39	0
10602	480	24	2	0	2024
24	276	24	1		

Row 6, d12^⊕2

337920	792576	1019904	903680	602112	313344
128976	41904	10596	2024	276	24
1	792576	1345024	1267200	800640	369024
127104	32040	5580	600	30	1019904
1267200	892480	409920	136560	34320	6225
720	40	903680	800640	409920	126080
26280	4140	480	30	0	602112
369024	136560	26280	2650	168	12
0	0	313344	127104	34320	4140
168	2	128976	32040	6225	480
12	41904	5580	720	30	0
10596	600	40	0	0	2024
30	276	24	1		

Row 7, e8^⊕3

175616	526848	790272	777728	555072	301056
126784	41664	10584	2024	276	24
1	526848	1185408	1317120	940800	470400
169344	43680	7728	840	42	790272
1317120	1086624	578592	214032	56448	10668
1344	84	777728	940800	578592	240632
71232	15792	2880	384	24	555072
470400	214032	71232	15792	2352	360
48	3	301056	169344	56448	15792
2352	84	126784	43680	10668	2880
360	41664	7728	1344	384	48

```
 10584    840     84     24      3   2024
    42    276     24      1
```

Row 8, d16 ⊕ e8

```
207872  566272  813568  785920  556672  301184
126784   41664   10584    2024     276      24
     1  566272 1152256 1254400  901504  456320
166560   43456    7728     840      42  813568
1254400 1074304  602112  231840   63168   12040
  1456      84  785920  901504  602112  256592
 71792   14168    2080     198       8  556672
456320  231840   71792   13076    1568     176
    16       1  301184  166560   63168   14168
  1568      56  126784   43456   12040    2080
   176   41664    7728    1456     198      16
 10584     840      84       8       1    2024
    42     276      24       1
```

Row 9, d24+

```
137216  362496  574464  619520  481536  278784
122496   41184   10560    2024     276      24
     1  362496  762368 1013760  887040  532224
221760   63360   11880    1320      66  574464
1013760 1182720  887040  443520  147840   31680
  3960     220  619520  887040  887040  554400
221760   55440    7920     495       0  481536
532224  443520  221760   66528   11088     792
     0       0  278784  221760  147840   55440
 11088     924  122496   63360   31680    7920
   792   41184   11880    3960     495       0
 10560    1320     220       0       0    2024
    66     276      24       1
```

Author information

Ibrahim Amin Nicola Shnoudi

Independent researcher, Amman, Jordan

Email: Ibrahim.shnoudiest@gmail.com

ORCID: 0009-0002-3706-6321

Correspondence concerning this paper, including reports of any error in the certificates of Sections 5 and 7 or in the data of Appendices B and C, may be addressed to the author at the above address.